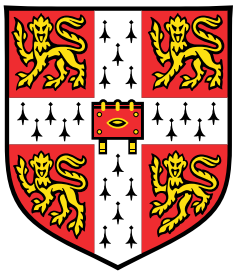
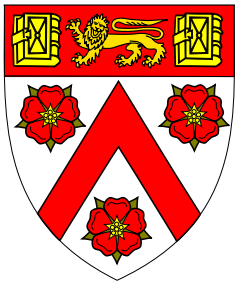


Deep Structured Multi-Task Learning for Computer Vision in Autonomous Driving



Marvin Teichmann

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Trinity College October 2019

I would like to dedicate this thesis to my loving parents for their endless support, encouragement and sacrifices.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university, except as declared in the Preface and specified in the text. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Marvin Teichmann
October 2019

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Roberto Cipolla for his constant support throughout my research, and for his motivation and immense knowledge. He has given me substantial room to grow and a lot of freedom in pursuing the various research ideas that sprang to my mind. I thank him for the wonderful opportunities he has opened up for me.

Secondly I would like to thank Trinity College for providing me with accommodation, funding and a welcoming and supportive community during my time in Cambridge. Special thanks are due to all the other members of the College for being such a great community, which has become like a second family to me in Cambridge.

I owe sincere thanks to Andre Araujo for hosting me as an intern at Google Research. I have benefited from our discussions and from his help, as well as his knowledge and experience in landmark recognition.

My warmest thanks also go to all the colleagues I was fortunate to learn from, and to discuss and collaborate with in producing the research in this thesis. In alphabetical order: Benjamin Biggs, Ignas Budvytis, James Charles, Je Hyeong Hong, Anthony Hu, Yani Ioannou, Fotis Logothetis, Roberto Mecca, Thomas Roddick and Tomas Vojir.

I am very grateful to Qualcomm Research Europe, for sponsoring my research with a generous, unconstrained stipend. Thanks also go to Paul Beverley for proofreading this thesis and providing helpful suggestions.

Very special thanks are reserved for Kristina Klein for all her love, support and encouragement, especially because, during the most stressful and difficult times, I could always rely on her moral and social support as well as her good advice. Neither this thesis nor my time in Cambridge would have been the same without her. She has my sincere thanks.

Abstract

The field of computer vision is currently dominated by deep learning advances. Convolutional Neural Networks (CNNs) have become the predominant tool for solving almost any computer vision task, so state-of-the-art systems have been built by using the predictive capabilities of Convolutional Neural Networks (CNNs). Many of those systems use simple encoder–decoder based design, where an off-the-shelf CNN architecture is combined with a task-specific decoder and loss function in order to create an end-to-end trainable model. This ultimately raises the question of whether these kinds of models are the future of computer vision.

In this thesis we argue that this is not the case. We start off by discussing three limitations of simple end-to-end training. We proceed by showing how it is possible to overcome those limitations by using an approach that we call structured modelling. The idea is to use CNNs to compute a rich semantic intermediate representation which is then used to solve the actual problem by applying a geometric and task-related structure.

In this work we solve the localization, segmentation and landmark recognition task using structured modelling, and we show that this approach can improve generalization, interpretability and robustness. We also discuss how this approach is particularly useful for real-time applications such as autonomous driving. Visual perception is a multi-module problem that requires several different computer vision tasks to be solved. We discuss how, by sharing computations, we can improve not only the inference speed but also the prediction performance by using the structural relationship between the tasks. Lastly, we demonstrate that structured modelling is able to achieve state-of-the-art performance, making it a very relevant approach for solving current and future computer vision problems.

Table of contents

List of figures

List of tables

List of Abbreviations

BMVC British Machine Vision Conference. 35

CNN Convolutional Neural Network. xiv, 7, 37, 39–42, 46–48, 55, 56

ConvCRF convolutional CRF. 22, 26, 30, 32

CRF Conditional Random Field. 21, 22

D2R detect-to-retrieve. xviii, xx, 56, 69–72, 74, 75

FCN Fully Convolutional Neural Network. 6

FullCRF fully connected CRF. 22–27, 30, 32

IV Intelligent Vehicles Symposium. 4

KIT Karlsruhe Institute of Technology. 4

PCA principal component analysis. 42

PnP perspective-n-point problem. xiv, 35, 37, 39, 43, 44

RANSAC Random sample consensus. 35, 43

RMAC regional Maximum Activations of Convolutions. xviii, 69, 70, 72, 74–76

RoI Region of Interest. 6, 9, 10

SIFT Scale-invariant feature transform. xvi, 54, 55, 57

UofT University of Toronto. 4

Chapter 1

Introduction

The field of computer vision is currently dominated by recent deep learning (Goodfellow et al., 2016) advances. Deep CNNs based models (Krizhevsky et al., 2012a) have proven to be a powerful tool which enables us to learn a hierarchy of rich semantic image features (Zeiler and Fergus, 2014) without explicit supervision. This has fuelled explosive advances in solving almost every computer vision task. For many of those tasks it is enough to take an off-the-self CNN architecture (He et al., 2016a; Howard et al., 2017; Simonyan and Zisserman, 2014a), add a suitable loss and train the model end to end (Kendall et al., 2015; Liu et al., 2016; Long et al., 2015). In many areas, this approach significantly outperforms the best traditional, non-deep learning based system (He et al., 2017a; Long et al., 2015; Ren et al., 2015a). This raises the question of whether we will be able to solve most if not all computer vision problems with a deep end-to-end trained network. Given enough data and computational power, is deep learning always the best method?

In this thesis we argue that this is not the case. Instead, we make the case for an approach that we call structured modelling. In this approach we use CNNs as feature extractors that are tasked to compute rich, interpretable representations. Those intermediate representations can then be used to solve the actual task by using more traditional algorithms. This approach allows us to incorporate prior knowledge about geometric or task-related structures. We show that this approach allows us to overcome the limitations of traditional end-to-end trained CNN models. We are able to produce more robust, interpretable models that are able to generalize better, given the same amount of data. Also, our method is able to keep up with, and even beat, the state of the art in several tasks.

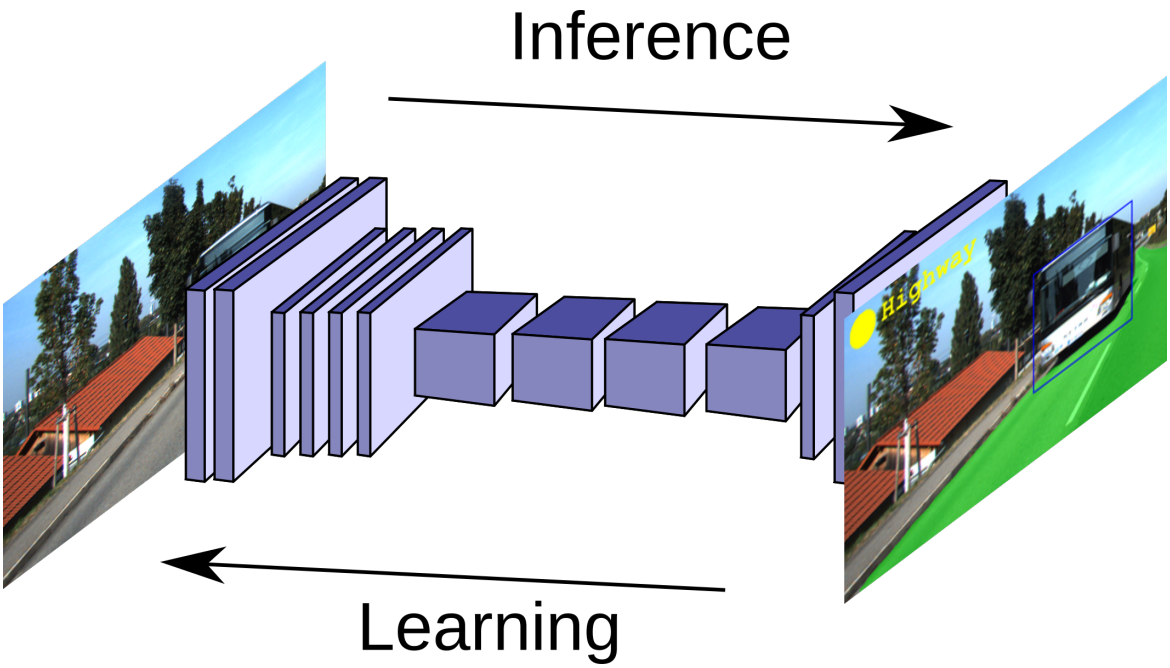


Fig. 1.1 Visualization of an end-to-end trainable CNN model. The network predicts a representation which can be easily transformed into the desired output. The network is trained to predict this representation directly.

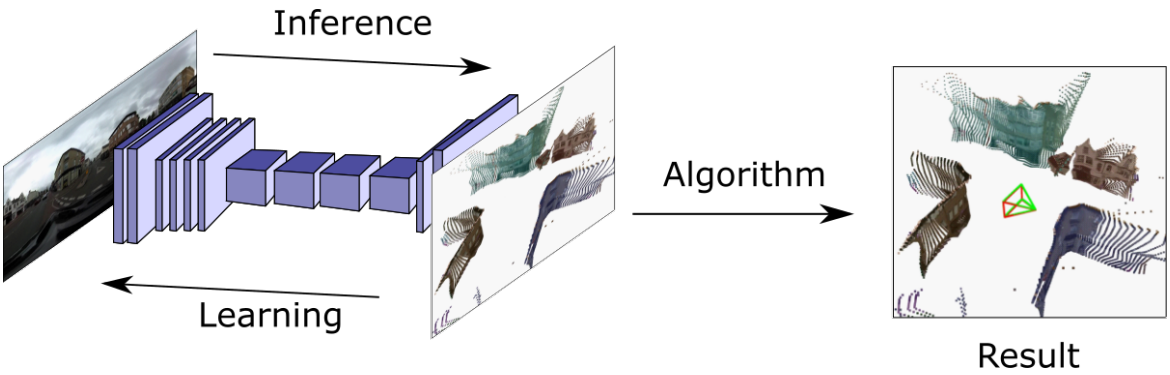


Fig. 1.2 Visualization of the structured modelling approach. The network is trained to predict an intermediate representation. We then utilize the structure of the representation to solve the actual task.

1.1 Structured Models

During my time as a student of computer vision deep end-to-end learning has been established as the dominant paradigm to solve vision tasks. Many tasks are solved by designing a CNN model whose output can easily be transformed into a solution of the task. The CNN can then be trained end-to-end with a suitable loss function as visualized in Figure 1.1. The method is very powerful since many vision tasks can be solved with it as long as it is possible to formulate the task such that each input has a unique and distinct output and it is possible to acquire suitable annotations for that.

In this thesis we well investigate how we can build models which are able to take advantage of geometric or task-related structure at inference time. In order to do so we propose to train CNN models to predict an interpretable intermediate representation. The representation can then be utilized to solve the actual task. Such a computational pipeline is visualized in Figure 1.2. In this thesis we will call such an approach a structured model. In literature such approaches are sometimes referred to as structure based models (Sattler et al., 2019a). We will show that building a structured pipeline can have many advantages over an end-to-end trained CNN including better generalization, interpretability and robustness.

1.2 Thesis Overview

The goal of this thesis is to promote structure-based deep learning methods which utilize task-specific structure in conjunction with representations computed by CNNs. This kind of explicit modelling has long been overlooked due to the unreasonable effectiveness of CNN-based end-to-end learning.

Towards this goal, this thesis starts by discussing three examples of the limitations of end-to-end learning approaches in Chapter 2 of which we are able to overcome using the structure-based methods proposed in succeeding chapters. In Chapter 3 we introduce ConvCRF a novel Conditional Random Field (CRF)-based model that is able to learn structure from data. In Chapter 4 we propose a novel structure-based camera localization method. The method is able to use geometric structure to generalize much better compared to an end-to-end trained baseline. The approach also adds interpretability and improves robustness. Finally, in Chapter 5 we use a structure-based model to perform landmark recognition. The model uses task-related structure to combine global reasoning with local feature descriptors. Doing this allows us to beat the state-of-the-art performance in a well established landmark recognition benchmark.

1.3 Contributions

In this thesis we show how structured-based approaches can improve upon simple end-to-end trained deep learning models. As part of this we make several novel contributions. The most important are:

- We propose a novel Conditional Random Field (CRF) design which is two orders of magnitude faster than comparable models and is able to learn structure from data.
- We propose a novel multi-task training approach, which is computationally efficient and very widely applicable.
- We show how structure base methods are able to overcome various limitations of classic end-to-end trained methods and achieve state-of-the-art performance in camera localization and landmark recognition tasks.

1.4 Co-Authored Publications

Most of the research presented in this thesis is based on work I have done in collaboration with other authors. This thesis extends work of the following four publications listed in chronological order:

- Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., and Urtasun, R. **Multinet: Real-time joint semantic reasoning for autonomous driving.** In *IEEE Intelligent Vehicles Symposium (IV)*, 2018. Cited over 200 times.
- Teichmann, M., Araujo, A., Zhu, M., and Sim, J. **Detect-to-retrieve: Efficient regional aggregation for image search.** In Proceedings of the *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- Budvytis, I., Teichmann, M., Vojir, T., and Cipolla, R. (2019). **Large scale joint semantic re-localisation and scene understanding via globally unique instance coordinate regression**. *British Machine Vision Conference (BMVC)*, 2019.
- Teichmann, M. and Cipolla, R. **Convolutional CRFs for semantic segmentation.** *British Machine Vision Conference (BMVC)*, 2019. Oral presentation.

In addition Section 2.3 extends work that I did at the University of Toronto (UofT) and the Karlsruhe Institute of Technology (KIT), and which was submitted as a masters thesis for my taught masters degree at KIT in 2016.

Chapter 2

The Limitations of End-to-End Learning

Deep end-to-end learning of all model features from data has been the powerhouse of machine learning in recent years. The widespread availability of data, computational resources and new optimization algorithms has enabled us to solve many once difficult computer vision problems by simply modelling them using an adequate loss function and training a deep network end to end. This ultimately raises the question whether deep end-to-end learning is the silver bullet which is able to solve all our problems, given the right loss function. In this chapter we discuss three examples which show the limitations of unstructured end-to-end learning.

2.1 CNN-based Camera Pose Regression

In this section we will discuss the limitations of unstructured end-to-end learning for the camera pose estimation task. The goal of pose estimation is to identify camera position and viewpoint angle used in a known environment, given an input image. An example of pose estimation is depicted in Figure 2.1.

As shown by Kendall et al. (2015), camera pose estimation can be trained end to end, using a regression loss. In such a model the network has the task of predicting a seven-dimensional vector. Three dimensions represent the predicted point in 3D and the other four the camera angle in quaternion representation. Other representations for the angle are possible, but don't change the model significantly.

In the field of autonomous driving, localisation in a known environment is a very important task. This task can be viewed as a pose estimation problem. In this setting, data is usually collected by driving along each road in the map and capturing a video sequence along with location information for any frame. For practical reasons, it is desirable to have a

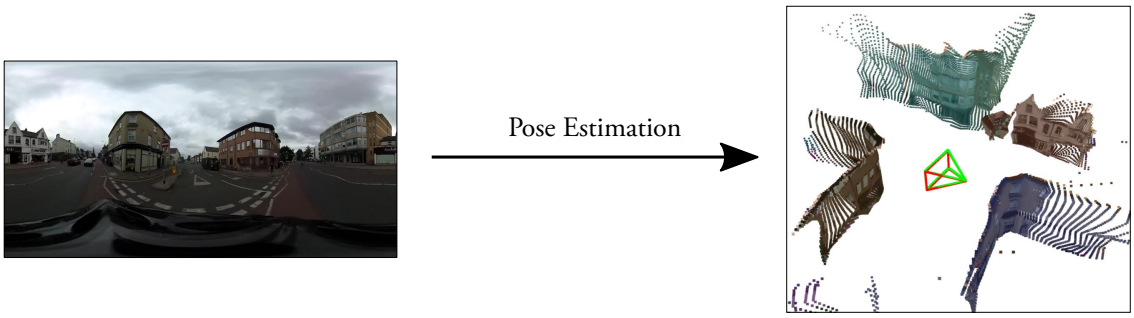


Fig. 2.1 Pose estimation task visualized. The goal of the pose estimate is to predict position and camera angle, given an image.

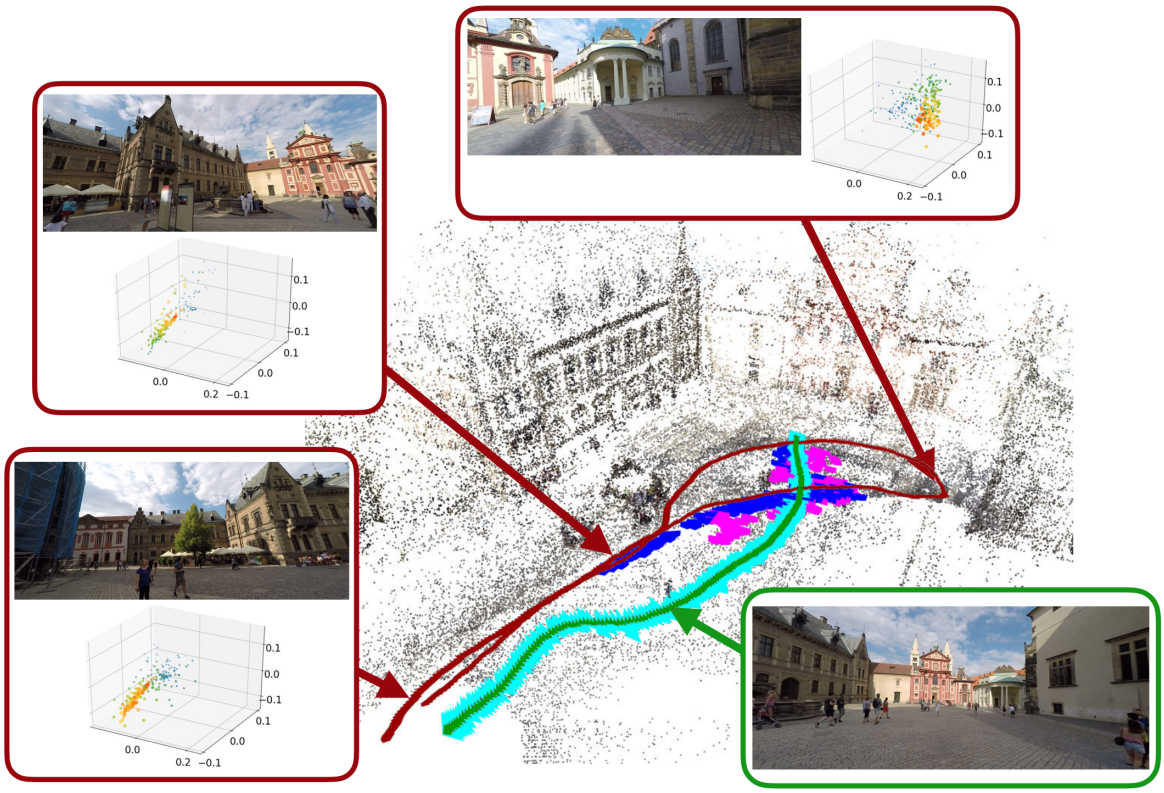


Fig. 2.2 Visualizing the generalization issue of PoseNet. Red shows the location of the training images, green of testing images. Blue indicates PoseNet predictions on testing images. It can be seen that PoseNet fails to generalize and predict positions close to the training track. Figure courtesy of Torsten Sattler.

system where it is sufficient to only driving through each road once. The goal is to accurately predict the current position when visiting the same area again at a later time.

In Cambridge, we did such localisation experiments using PoseNet (Kendall et al., 2015). Data was collected by a camera mounted on a person's helmet. We rode through each street once to collect training data and a second time to obtain test images. We observed that PoseNet is able to predict the locations on test data reasonable well. However, it will always place its predicted position on the trajectory of the training track, even in cases where the exact path taken by train and test rides differ. This effect is visualized in Figure 2.2. In addition, PoseNet would fail to generalize the camera angle and predict an angle similar to the one given in the training track. The same finding even holds true in experiments where we rode on the opposite side of the street.

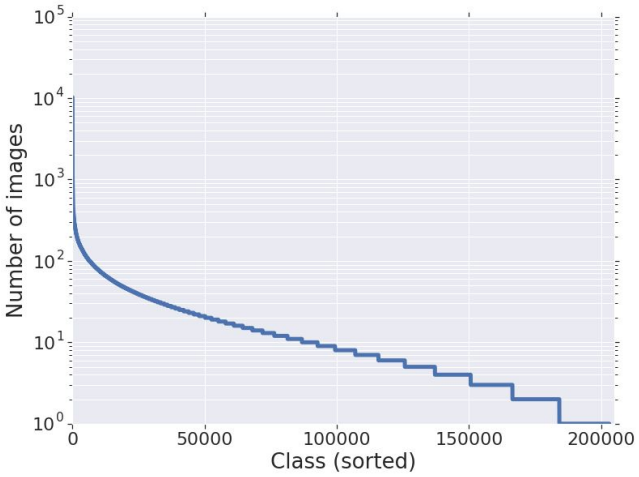
Sattler et al. (2019b) did an in-depth analysis of end-to-end trained pose regression systems like PoseNet (Kendall et al., 2015). They developed a theory about pose regression which indicates that those systems behave very similarly to image-retrieval baselines. Given a test image, PoseNet would output a weighted average of positions from similar-looking training images, according to that theory. Sattler et al. (2019b) also provided experimental evidence that their theory is correct.

In Chapter 4 we will show how we can overcome these limitations by using geometric structure. Instead of predicting the camera position directly, our approach predicts the entire 3D mesh of the surrounding world and then solves a perspective-n-point problem (Kneip et al., 2014). Such an approach has no issue with generalizing across trajectories or viewpoint angles. Our approach is even able to correctly predict location and angle when cycling on the opposite side of the road in the opposite direction to the training track.

2.2 Landmark Recognition and Long-Tailed Classification

Landmark recognition is the task of identifying an entity of interest ("Landmark"). Landmark recognition can be viewed as a classification task, since one categorical prediction per image is made. However, in practice there are two major differences between typical classification datasets and landmark recognition datasets. Landmark recognition tasks have many more classes and the distribution of examples per class is very long tailed.

Consider the Google Landmark dataset (Bohyung Han, 2019). It consists of more then 5 million images containing over 200 000 unique landmarks. This is much larger than typical classification datasets. In addition, the distribution of the examples per class is very long tailed. Consider Figure 2.3, almost half of the landmarks are shown in fewer than ten images, and one in ten landmarks are only shown in one image. For such a dataset a standard



Number of images	Proportion of classes with less or equal images
1	9%
5	38%
10	57%
25	80%
100	93%

Fig. 2.3 Number of images for each landmark. It can be seen that the distribution is very long-tailed. Almost one in ten landmarks are only shown in one image, and half of the landmarks are shown in less then ten images. However, there are some landmarks with up to 10 000 examples.

classification loss, which treats each class as a separate entity, is not suitable. In chapter 5 we discuss systems that predict the keypoint and perform geometric matching to overcome this limitation.

A system solving landmark recognition should be able to use the wide range of classes to generalize discriminators between classes. In addition, the system should be able to solve the one and ideally zero shot classification problem. Both of those issues are not solved when training a classification network end to end.

2.3 MultiNet: Multi-Task Learning for Autonomous Driving

In this section we introduce MultiNet, a novel encoder-based end-to-end learnable multi-tasking approach. MultiNet is able to jointly solve semantic segmentation, detection and classification in one network. All three tasks are highly relevant to autonomous driving as visualized in Figure 2.4. Our method is able to benefit from shared computation in order to improve inference efficiency while maintaining state-of-the-art performance in the individual tasks.

While our method can improve in terms of inference speed, the performance improvement in term of prediction accuracy is negligible compared to the individual tasks. This is despite very favourable conditions for co-adaptations between tasks. This indicates that simple

Fig. 2.4 Our goal: Solving street classification, vehicle detection and road segmentation in one forward pass.

unstructured feed-forward learning might not be enough to archive the goal that multiple tasks benefit from each other.

This chapter is largely based on my paper *MultiNet: Real-time joint semantic reasoning for autonomous driving* (Teichmann et al., 2018) which was published at the Intelligent Vehicles Symposium (IV) 2018 and won the *Best Poster Paper Award* at the IV. The discussed methods and implementation got a lot of attention in the research community, and was cited more than 200 times in follow-up work. Some ideas and methods of the paper are based on work I did at the University of Toronto (UofT) and the Karlsruhe Institute of Technology (KIT), and which was submitted as a masters project for my taught masters degree at KIT in 2016.

2.3.1 Background

While most approaches to semantic reasoning have focused on improving performance, in this section we argue that computational times are very important in order to enable real-time applications such as autonomous driving. Towards this goal, we present an approach to joint classification, detection and semantic segmentation using a unified architecture where the encoder is shared between the three tasks. Our approach is very simple, can be trained end-to-end and performs extremely well in the challenging KITTI dataset. Our approach is also very efficient, allowing us to perform inference at more then 23 frames per second.

Current advances in the field of computer vision have made clear that visual perception is going to play a key role in the development of self-driving cars. This is mostly due to the deep learning revolution which began with the introduction of AlexNet in 2012 by Krizhevsky et al. (2012b). Since then, the accuracy of new approaches has been increasing rapidly. The causes for this are the existence of more data, increased computational power and developments in

algorithms. The current trend is to create deeper or wider networks with as many parameters as possible (He et al., 2015a; Huang et al., 2017a; Wu et al., 2019).

While performance is already extremely high, when dealing with real-world applications, running time becomes important. New hardware accelerators, as well as compression, reduced precision and distillation methods have been exploited to speed up current networks.

In this section we take an alternative approach and design a network architecture that can very efficiently perform classification, detection and semantic segmentation simultaneously. This is done by incorporating all three tasks into a unified encoder–decoder architecture. We name our approach MultiNet. A single neural network predicts solutions for classification, detection and segmentation tasks in one evaluation. To achieve this, we propose an encoder–decoder design.

The encoder is a deep CNN, producing rich features that are shared between all the tasks. These features are then used by task-specific decoders, which produce their outputs in real time. Each decoder is independently adjusted for its corresponding task. We demonstrate the effectiveness of our approach in the challenging KITTI benchmark (Geiger et al., 2012) and show state-of-the-art prediction performance. In addition, our approach benefits from sharing computations, which makes it computationally very efficient. This benefits the inference speed greatly, allowing us to solve all three tasks in less than 45ms.

Contribution: Our main contribution is three-fold: Firstly we introduce a novel training method for multi-tasking learning, which allows our model to use independent hyperparameters for each task. Secondly we produce state-of-the-art results with novel approaches for all three tasks, which are relevant for autonomous driving. Lastly we introduce a new multi-tasking architecture which is computationally very efficient and has a fast, real-time-capable inference speed.

2.3.2 Related Work

In this section we review current approaches to the tasks that MultiNet tackles, i.e. detection, classification and semantic segmentation. We focus our attention on deep learning based approaches.

Classification After the development of AlexNet by Krizhevsky et al. (2012b), most modern approaches to image classification use deep learning. Residual networks (He et al., 2015a) are the state of the art, because they allow us to train very deep networks without problems of vanishing or exploding gradients. In the context of road classification, deep neural networks are also widely employed (Ma et al., 2016). Sensor fusion has also been

exploited in this context (Seeger et al., 2016). In this paper we use classification to guide other semantic tasks, i.e. segmentation and detection.

Detection Traditional deep learning approaches to object detection follow a two-step process, where region proposals (Hosang et al., 2015, 2014; Lampert et al., 2008) are first generated and then scored using a convolutional network (Girshick et al., 2013; Ren et al., 2015b). Additional performance improvements can be gained by using convolutional neural networks (CNNs) for the proposal generation step, as proposed by Erhan et al. (2013); Ren et al. (2015b) or by reasoning in 3D (Chen et al., 2016b, 2015b). Recently, several methods have proposed to use a single deep network that is trainable end-to-end to directly perform detection (Liu et al., 2015a,a; Sermanet et al., 2013; Stewart et al., 2016). Their main advantage over proposal-based methods is that they are much faster at both training and inference time, and thus more suitable for real-time detection applications. However, they lag far behind in performance. In this chapter we propose an end-to-end trainable detector that reduces significantly the performance gap. We argue that the main advantage of proposal-based methods is their ability to have size-adjustable features. This inspired our Region of Interest (RoI) pooling implementation.

Segmentation Inspired by the successes of deep learning, CNN-based classifiers were adapted to the task of semantic segmentation. Early approaches used the inherent efficiency of CNNs to implement implicit sliding-window (Giusti et al., 2013; Li et al., 2014). A breakthrough was archived by Long et al. (2015) with the introduction of *Fully Convolutional Neural Networks (FCNs)*, which allow us to model semantic segmentation using a deep learning pipeline that is trainable end-to-end. Transposed convolutions (Dumoulin and Visin, 2016; Im et al., 2016; Zeiler et al., 2010) are used to up-sample low resolution features. A variety of deeper flavours of FCNs have since been proposed Badrinarayanan et al. (2015); Noh et al. (2015); Papandreou et al. (2015); Ronneberger et al. (2015a). Very good results are achieved by combining FCN with conditional random fields (CRFs) (Chen et al., 2014, 2016a; Zheng et al., 2015b). Schwing and Urtasun (2015) and Zheng et al. (2015b) showed that mean-field inference in the CRF can be cast as a recurrent net, allowing end-to-end training. Dilated convolutions were introduced by Yu and Koltun (2015a) to augment the receptive field size without losing resolution.

These techniques, in conjunction with residual networks He et al. (2015a), were state-of-the-art when our method was proposed. In more recent years, pyramid-pooling Chen et al. (2017a); Zhao et al. (2017) has been proposed as a way of introducing global context into the model.

Multi-task learning Multi-task learning techniques aim at learning better representations by exploiting many tasks. Several approaches have been proposed in the context of CNNs: (Liu et al., 2015c; Long and Wang, 2015). An important application for multi-task learning is face recognition (Ranjan et al., 2016; Yim et al., 2015; Zhang et al., 2014).

Learning semantic segmentation, in order to perform detection or instance segmentation has been studied (Dai et al., 2016; Gidaris and Komodakis, 2015; Pinheiro et al., 2016). In those systems, the main goal is to perform an instance level task. Semantic annotation is only viewed as an intermediate result. Systems like (Sermanet et al., 2013; Wu et al., 2016) and many more design one system which can be fine-tuned to perform tasks like classification, detection or semantic segmentation. In this kind of approach, a different set of parameters is learned for each task. Thus, joint inference is not possible in this models. The system described in Hariharan et al. (2014) is closest to our model. However, this system relies on existing object detectors and does not fully leverage the rich features learned during segmentation for both tasks. To the best of our knowledge our system is the first one proposed that is able to do this.

Many multi-tasking systems for visual perception have been introduced following the initial presentation of our work (Chen et al., 2017b; Kendall et al., 2018; Radwan et al., 2018). Many follow an encoder–decoder based approached, as proposed in our work (Kendall et al., 2018; Xiao et al., 2018; Xu et al., 2018).

2.3.3 MultiNet for Joint Semantic Reasoning

In this section we propose an efficient and effective feed-forward architecture, which we call *MultiNet*, to jointly reason about semantic segmentation, image classification and object detection. Our approach shares a common encoder over the three tasks and has three branches, each implementing a decoder for a given task. Fig. 2.5 gives an illustration of our architecture. MultiNet can be trained end-to-end, and joint inference over all tasks can be done in less than 45ms. We start our discussion by introducing our joint encoder, followed by the task-specific decoders.

Encoder

The task of the encoder is to process the image and extract rich abstract features (Zeiler and Fergus, 2014) that contain all the necessary information to perform accurate segmentation, detection and image classification. The encoder consists of the convolutional and pooling layers of a classification network. The weights of the encoder are initialized using the

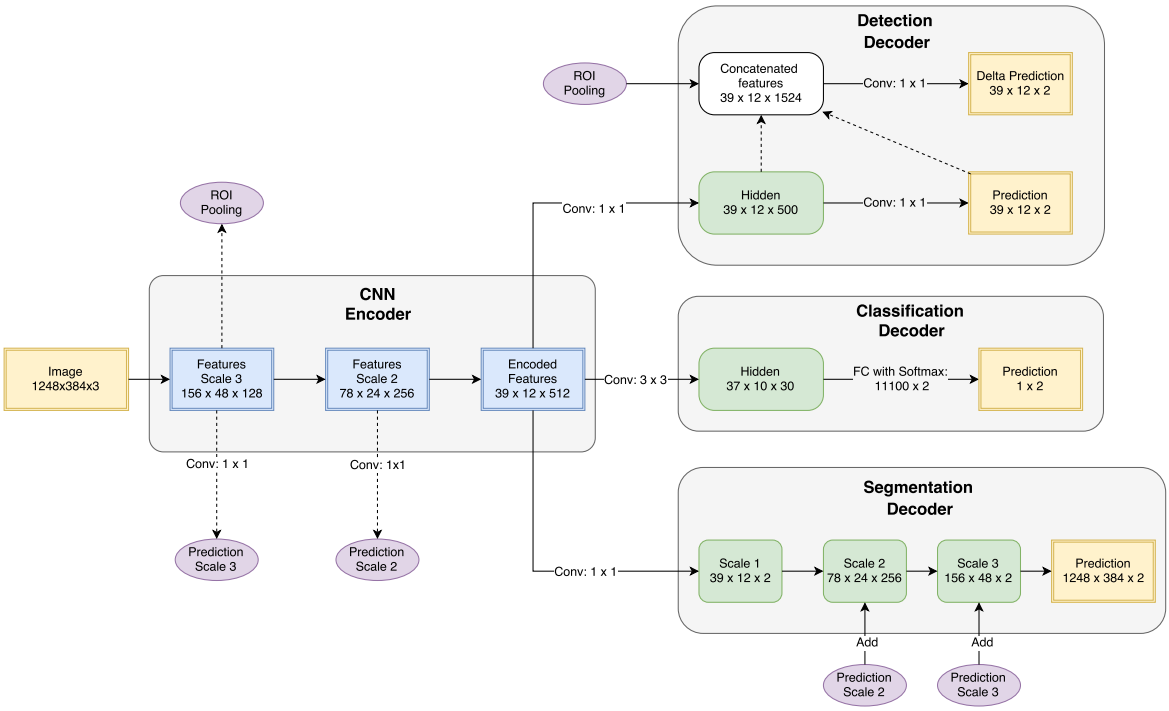


Fig. 2.5 MultiNet architecture.

weights pre-trained on ImageNet Classification Data (Russakovsky et al., 2015). Any modern classification network can be used as an encoder.

We perform experiments using versions of VGG16 (Zeiler and Fergus, 2014) and ResNet (He et al., 2015a) architectures. Our first VGG encoder uses all the convolutional and pooling layers of VGG16, but discards the fully connected softmax layers. We call this version *VGG-pool5*, because pool5 is the last layer used from VGG16. The second implementation only discards the final fully connected softmax layer. We call this architecture *VGG-fc7*, because fc7 is the last layer used from VGG16. VGG-fc7 uses two fully connected layers from VGG: *fc6* and *fc7*. We replace those layers with equal 1×1 convolutions as discussed in (Long et al., 2015; Sermanet et al., 2013). This idea allows the encoder to process images with arbitrary input size. In particular, we are not bound to the original VGG input of 224×224, which would be too small to perform perception in street scenes.

For ResNet we implement the 50 and 101 layer version of the Network. For the encoder, we use all the layers apart from the layers of fully connected softmax.

Classification Decoder

We implement two classification decoders. One version is a vanilla fully connected layer with softmax activation. This encoder is used in conjunction with an input size of 224×224.

Thus the overall network is equal to the original VGG or ResNet respectively, when used with the corresponding encoder. The purpose of this encoder is to serve as high quality baseline to show the effectiveness of our scene classification approach. This first classification encoder cannot be used for joint inference with segmentation and detection. Both approaches require a larger input size. Increasing the input size on this classification encoder, however, yields far too many parameters for the final layer.

The second classification decoder is designed to take advantage of the high resolution features that our encoder generates. In typical image classification tasks (e.g. (Krizhevsky et al., 2016; Russakovsky et al., 2015)) the input features one object, usually centred prominently in the image. For this kind of task it is reasonable to use a very small input size. Street scenes on the other hand contain a large number of smaller scale objects. We argue that it is vital to use high resolution input in order to use the features that those objects provide. By increasing the input size of our image to 1248×348, we effectively apply our feature generator to each spatial location of the image (Long et al., 2015; Sermanet et al., 2013). The result is a grid of 39×12 features, each corresponding to a spatial region of 32×32 pixels. In order to utilize these features, we first apply a 1×1 convolution with 30 channels. This layer serves as a *BottleNeck*. The main purpose is to greatly reduce dimensionality.

Detection Decoder

The detection decoder is designed to be a proposal-free approach similar to ReInspect (Stewart et al., 2016), Yolo (Redmon et al., 2015) and Overfeat (Sermanet et al., 2013). Omitting an artificial proposal generator step produces much faster inference. This is crucial for our goal of building a real-time-capable detection system.

We do however observe that proposal-based detection systems outperform proposal-free approaches. We argue that this performance gap is largely explained by the fact that proposel-based detection systems are able to incooperate rescaling into their network. This makes the model invariant to scale, a characteristic CNNs commonly lack. Region of Interest (RoI) pooling (Girshick, 2015) and RoI align (He et al., 2017b) are commonly used methods to obtain scale invariance.

Our goal is to integrate a rescaling layer similar to RoI pooling into a fast proposal free approach. Towards this goal we first design a detection decoder similar to Yolo (Redmon et al., 2015). The decoder outputs a grid of cells $\mathscr{G}_c$ with a fixed shape of 39×12 as illustrated in Figure 2.6. Each cell corresponds to a 32×32 area in the image, those exact dimensions are obtained by the internal down-sampling of the network. For each cell we predict two information. Firstly whether an object is present at the location of the cell and secondly the bounding box of the object which is at the location of the cell. It is assumed that all

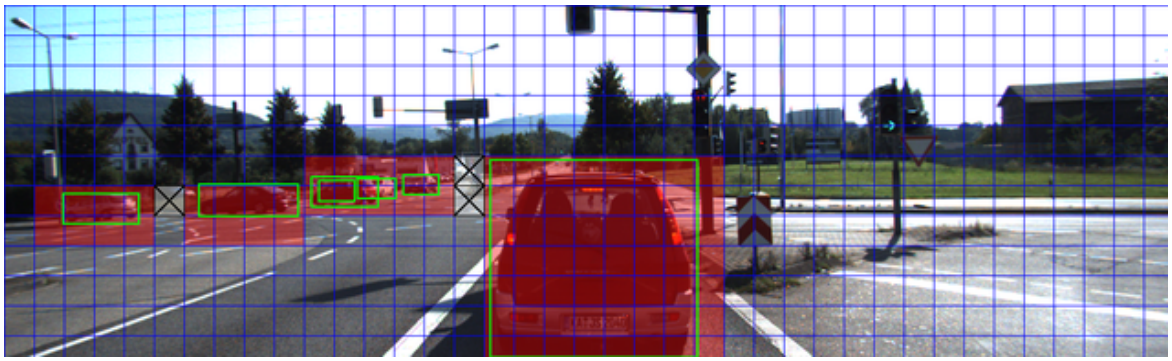


Fig. 2.6 Visualization of our detection encoding. Blue grid: cells, Red cells: cells with positive confidence label. Transparent cells: cells with negative confidence label. Grey cells: cells in a 'don't care area'. Green boxes: ground truth boxes.

relevant objects have a size of at least 32 pixels in both spatial dimension such that each object intersects with multiple cells. Each cell predicts exactly one bounding box. If the first prediction is negative, i.e. their is no object present at the location of the cell, then the predicted bounding box is simply ignored. If multiple bounding boxes intersect with a cell the cell is asked to predict the bounding box of the object whos center is closest to the center of the cell.

In order to integrate rescaling into this approach we utilize the fact that each cell predicts exactly one box. Given a cell c we take the predicted bounding box B_c and we apply adaptive average pooling over the corresponding area in the feature map from earlier outputs of the encoder. In other words we average over the feature vectors of all cells within the area of B_c and obtain a new feature vector f_c. The resulting grid $\mathscr{G}_F$ of feature vectors has the same spatial dimensions (39×12) as $\mathscr{G}_c$. In $\mathscr{G}_c$ each cell correspond to a fixed area of 32×32 pixels. In $\mathscr{G}_F$ each cell corresponds to a dynamically sized area potentially much larger than 32 pixels.

For the final prediction we concatinate $\mathscr{G}_F$ with the non-averaged feature vectors and then apply a 1×1 convolutional to obtain new predictions p_2. For the second prediction we do however only predict the residuals. That is the final prediction is given as $p_f = p_1 + p_2$, where p_1 is the prediction corresponding to the first grid $\mathscr{G}_c$. The reason we obted for residual prediction is that the feature map $\mathscr{G}_F$ represents the image with distorted fashion. With those feature maps it is much easier for the network to identify an offset than it is to perform a completly new coordinate regression.

Segmentation Decoder

The segmentation decoder follows the main ideas of the FCN architecture (Long et al., 2015). Given the features produced by the encoder, we produce a low resolution segmentation of size 39×12 using a 1×1 convolutional layer. This output is then up-sampled using three transposed convolution layers (Dumoulin and Visin, 2016). Skip connections are used to extract high resolution features from the lower layers. Those features are first processed by a 1×1 convolution layer and then added to the partially up-sampled results.

2.3.4 Loss functions

Classification and segmentation are trained using a softmax cross-entropy loss function.

For the detection, the final prediction is a grid of 12×39 cells. Each cell gets assigned a confidence label as well as a box label. The box label encodes the coordinates of the box and is parametrized relative to the position of a cell. A cell c gets assigned a positive confidence label if and only if it intersects with at least one bounding box. If this is the case then the cell also gets assigned to predict the coordinates of the box with which it intersects. If multiple boxes intersect with a cell, the box whose centre is closest to the centre of c is chosen. Note that one box can be predicted by multiple cells.

If a box b is assigned to a cell c the following values are stored in c:

$$c_x = (x_b - x_c)/w_c \qquad c_y = (y_b - y_c)/h_c \tag{2.1}$$

$$c_w = w_b/w_c \qquad c_h = h_b/h_c \tag{2.2}$$

where x_b, y_b and x_c, y_c correspond to the centre coordinates of b and c, and w and h denote width and hight. Note, that w_c and h_c are always 32, as the cells of our model have a fixed width and height. We use L1 as our loss

$$\text{loss}_{\text{cell}}(c, \hat{c}) \;:=\; \delta c_p \cdot (|c_x - \hat{c}_x| + |c_y - \hat{c}_y| + |c_w - \hat{c}_w| + |c_w - \hat{c}_w|) \tag{2.3}$$

where $\hat{c}$ is the prediction of a cell and c its ground-truth, and c_p denotes whether a positive label has been assigned to a cell. The δc_p term ensures that the regression loss is zero if no object is present. We train the confidence labels using cross-entropy loss. The loss per cell is given as the weighted sum over the confidence and the regression loss. The loss per image is the mean over the losses of all cells. The KITTI Dataset also contains 'don't care areas'. Those areas are handled by multiplying the loss of the corresponding cells with zero. We note, that our label representation is much simpler than Faster-RCNN or ReInspect. This is

an additional feature of our detection system. The loss for MultiNet is given as the sum of the losses for segmentation, detection and classification.

2.3.5 Training Details

In this section, we describe the loss functions that we employ, as well as other details of our training procedure including initialization.

Initialization The weights of the encoder are initialized using weights trained on ImageNet (Deng et al., 2009a) data. The weights of the detection and classification decoder are initialized using the initialization scheme of (He et al., 2015b). The transposed convolution layers of the segmentation decoder are initialized to perform bilinear up-sampling. The skip connections of the segmentation decoder are initialized to very small weights. Both these modifications greatly improve segmentation performance.

MultiNet training strategy MultiNet training follows a fine-tuning approach. First, the encoder network is trained to perform classification on the ILSVRC2012 (Deng et al., 2009a) data. In practice, this step is omitted. Instead, we initialize the weights of all layers of the encoder with weights published by the authors whose network architecture we are using.

In a second step, the final fully connected layers are removed and replaced by our decoders. Then the network is trained end-to-end using KITTI data. Thus MultiNet training follows a classic fine-tuning pipeline.

Our joint training uses *asynchronous stochastic gradient descent* (Zhang et al., 2013) to update weights for each task (semi-)independently. In our implementation the forward passes for each of the tasks are computed independently in their own batch. The gradients are only added during the back-propagation steps. This is similar to the idea of using asynchronous stochastic gradient descent to speed up multi-gpu training (Lian et al., 2015; Paine et al., 2013). The difference is that we operate asynchronously across tasks and not necessarily across GPUs.

Computing those gradients asynchronously has several practical and theoretical advantages. It allows us to use different hyper-parameters for each task. A one-size-fits-all training does not work for all tasks. Segmentation training works best with a lower batch size and learning rate than does classification. In addition, classification needs more aggressive data-augmentation than the segmentation task to perform well. Only by having the freedom of choosing the optimal hyper-parameters for each task we can hope to get similar performance for multi-tasking and for individual task training.

The novel training method of MultiNet is one of the main contributions of our work (Teichmann et al., 2018) and is partly responsible for the success of the paper.

Loss Balancing The detection decoder uses two losses. A softmax cross-entropy loss to predict which cells contain cars (as visualized in Figure 2.6) and an L1 loss to regress the box coordinates. To balance those two losses a weight of 0.17 is applied to the L1 loss. The weight is chosen using a grid search, minimizing detection error on the validation set.

For the multi-task training the loss is balanced by appropriately normalizing the softmax cross-entropy loss. For each task the softmax cross-entropy loss is divided by the number of label signals present. For the classification task this corresponds to batch size, for detection to the total number of cells in the batch and for segmentation to the number pixels. This normalization ensures that all three losses are within the same order of magnitude. In addition all three normalized cross-entropy losses are guarantied to be between 0 and 1. In order to balance the cross-entropy losses with the L1 loss we apply the weight of 0.17 to the L1 loss. The grid search for the detection decoder indicates that this weight is a good balance between cross-entropy and L1. Our experiments indicate that our normalization approach is able to provide a good balance between the losses. We are able to reproduce the individual performance of each decoder in a multi-task setting. We have also tried to introduce an additional weight to each of the decoder but found that this does not yield a significant performance boost.

Optimizer and regularization We use the Adam optimizer (Kingma and Ba, 2014a) with a learning rate of 10^{-5} to train our MultiNet. A weight decay of $5 \cdot 10^{-4}$ is applied to all layers and dropout with probability 0.5 is applied to the 3×3 convolution of the classification and all 1×1 convolutions of the detection decoder.

We use a batch size of one for segmentation, five for detection and 16 for classification. For segmentation we choose a batch size of one so that we can train the model with differently size images. For detection and classification the batch size is chosen such that the resulting model fits comfortably into the memory of our GPUs.

Standard data augmentation is applied to increase the amount of effective available training data. We augment colour features by applying random brightness and random contrast. Spatial features are distorted by applying random flip, random resize and random crop.

2.3.6 Experimental Results

In this section we perform our experimental evaluation on the challenging KITTI dataset.

Fig. 2.7 Visualization of the segmentation output. Top row: Soft segmentation output as red and blue plot. The intensity of the plot reflects the confidence. Bottom row: Hard class labels.

Dataset

We evaluate MultiNet on the KITTI Vision Benchmark Suite (Geiger et al., 2013). The Benchmark contains images showing a variety of street situations captured from a moving platform driving around the city of Karlsruhe. In addition to the raw data, KITTI comes with a number of labels for different tasks relevant to autonomous driving. We use the road benchmark of (Fritsch et al., 2013) to evaluate the performance of our semantic segmentation decoder, which provides 289 annotated images. To train the object detection benchmark we use the data provided by (Geiger et al., 2012), which consists of 7000 annotated images. We exploit the automatically generated labels of (Ma et al., 2016), which provide us with road labels generated by combining GPS information with open-street map data. We obtain labels for all 7000 images of the detection set.

Overall, we have more than an order of magnitude more data for the classification and detection tasks, the additional data from them should be able to improve the generalization of the segmentation network in the multi-task setup considerably.

Detection performance is measured using the average precision score (Everingham et al., 2012). For evaluation, objects are divided into three categories: easy, moderate and hard to detect. The segmentation performance is measured using the MaxF1 score (Fritsch et al., 2013). In addition, the average precision score is given for reference. Classification performance is evaluated by computing the mean accuracy, precision and recall.

Experimental Evaluation

This section is structured as follows. We first evaluate the performance of the three decoders individually. To do this we fine tune the encoder using just one of the three losses – segmentation, detection and classification – and compare their performance with a variety of baseline. In the second part, we compare joint training of all three decoders with individual inference and show that the performance of joint training can keep up with the performance

Method	MaxF1	AP	Place
FTP Laddha et al. (2016)	91.61 %	90.96 %	6th
DDN Mohan (2014)	93.43 %	89.67 %	5th
Up_Conv_Poly Oliveira et al. (2016)	93.83 %	90.47 %	4rd
DEEP-DIG Muñoz-Bulnes et al. (2017)	93.83 %	90.47 %	3th
LoDNN Caltagirone et al. (2017)	94.07 %	92.03 %	2rd
MultiNet	**94.88 %**	**93.71 %**	1st

Table 2.1 Summary of the URBAN ROAD scores on the public KITTIRoad Detection Leaderboard Geiger (2013) at submission time.

Task: Metric	MaxF1	AP
VGG-pool5	95.80 %	92.19 %
ResNet50	95.89 %	92.10 %
VGG-fc7	95.94 %	92.24 %
ResNet101	96.29 %	92.32 %

(a) Prediction scores of our segmentation.

	speed [msec]	speed [fps]
VGG-pool5	42.14 ms	23.73 Hz
ResNet50	39.56 ms	25.27 Hz
VGG-fc7	96.84 ms	10.32 Hz
ResNet101	69.91 ms	14.30 Hz

(b) Inference speed of our segmentation.

Table 2.2 Performance of our segmentation approach.

of individual inferences. Overall we show, that our approach is competitive with individual inference. This makes our approach very relevant. Joint training has many advantages in robotics application, such as a fast inference time.

Segmentation The segmentation decoder–encoder is trained using the four different encoders discussed in Section 2.3.3. The scores, computed on a halt-out validation set, is reported in Table 2.2.

To compare our approach against the state of the art, we trained a segmentation network with the VGG-fc7 encoder on the whole training set and submitted the results to the KITTI road leaderboard. At submission time, our approach achieved first place in the benchmark. Recently our approach was overtaken by newer submissions. Some of those are directly based on our work and utilize our code Siam et al. (2017). The results at the time of our benchmark submission are given in Table 2.1.

The qualitative results are shown in Fig. 2.7 both as red and blue plots showing the confidence level at each pixel, as well as a hard prediction using a threshold of 0.5.

Detection The detection decoder is trained and evaluated on the data provided by the KITTI object benchmark Geiger et al. (2012). We train the detection decoder on a VGG

Task: Metric	moderate	easy	hard
VGG no RIO pool	77.00 %	86.45 %	60.82 %
Faster-RCNN	78.42 %	91.62 %	66.85 %
VGG-pool5	84.76 %	92.18 %	68.23 %
ResNet50	86.63 %	95.55 %	74.61 %
ResNet101	89.79 %	96.13 %	77.65 %

Table 2.3 Performance of our detection decoder.

	speed [msec]	speed [fps]	processing
VGG no RIO	35.75 ms	27.96 Hz	2.46 ms
Faster-RCNN	78.42 ms	12.75 Hz	5.3 ms
VGG-pool5	37.31 ms	26.79 Hz	3.61 ms
ResNet50	40.09 ms	24.93 Hz	3.19 ms
ResNet101	65.89 ms	15.17 ms	3.11 ms

Table 2.4 Inference speed of our detection detection.

Simonyan and Zisserman (2014b) and ResNet He et al. (2015a) decoder and evaluate on a validation set. Table 2.3 shows the results of our decoder compared to a Faster-RCNN baseline, evaluated on the same validation set. The results show that our rescaling approach is very efficient. Training the detection decoder with rescaling is only marginally slower than training it without. However, it offers a significant improvement in detection performance. Overall, our approach achieves a speed-up compared to Faster-RCNN of almost a factor of 2 and outperforms its detection accuracy. Qualitative results of the detection decoder can be seen in 2.8.

All in all, our results indicate that using a rescaling layer in order to achieve scale invariance is a good idea. A rescaling layer might be the key to closing the gap between proposal- and non-proposal-based approaches.

Our detection decoder is trained and evaluated on the data provided by the KITTI object benchmark Geiger et al. (2012). We train our detection decoder on a VGG Simonyan and Zisserman (2014b) and ResNet He et al. (2015a) decoder and evaluate on a validation set. Table 2.3 shows the results of our decoder compared to a Faster-RCNN baseline, evaluated on the same validation set. We report the inference speed in Table 2.4. We observe that our approach achieves a speed-up over Faster-RCNN of almost a factor of 2 and outperforms its detection accuracy. This makes our decoder particularly suitable for real-time applications. Qualitative results of our detection decoder can be seen in 2.8.

Fig. 2.8 Visualization of the detection output. With and without non-maximal suppression applied.

	mean Acc.	Precision	Recall
VGG-pool5 [our]	97.34 %	98.52 %	87.58 %
ResNet50 [our]	98.86 %	100.00 %	94.11 %
ResNet101 [our]	99.84 %	98.70 %	100.00 %
VGG16 [base]	93.04 %	91.61 %	87.90 %
ResNet101 [base]	93.83 %	91.94 %	89.54 %

Table 2.5 Classification performance of our decoder compared to baseline classification.

Classification The classification data is not part of the official KITTI Benchmark. To evaluate the classification decoder we first need to create our own dataset. This is done using the method described in Ma et al. (2016). To obtain a meaningful task, all images of one scene either fully in the train or fully in the validation set. This is important because the images of one scene are usually visually very similar.

We use a vanilla ResNet and VGG classification approach as baseline and compare this to a VGG and ResNet approach with our classification decoder. The differences between those two approaches are discussed in more detail in Section 2.3.3. The results are reported in Table 2.5 and Table 2.6. Our customized classification decoder clearly outperforms vanilla decoders, showing the effectiveness of our approach.

MultiNet We ran a series of experiments comparing VGG and ResNet as encoder. Table 2.7 and Table 2.8 compare performance of VGG and ResNet. We observe that both ResNet-

	speed [msec]	speed [fps]
VGG-pool5 [our]	37.83 ms	26.43 Hz
ResNet50 [our]	44.27 ms	27.96 Hz
ResNet101 [our]	71.62 ms	22.58 Hz
VGG16 [base]	7.10 ms	140 Hz
ResNet101 [base]	33.06 ms	30.24 Hz

Table 2.6 Inference speed of our classification.

based encoders are able to outperform VGG slightly, although there is a trade-off, as the VGG encoder is faster.

The speed gap between VGG-pool5 and ResNet50 is much larger when performing joint inference, compared to the individual task. This can be explained due to the fact that ResNet computes features with 2048 channels, while VGG features have only 512 channels. Thus, computing the first layer of each decoder is significantly more expensive.

Overall we conclude that MultiNet using a VGG decoder offers a very good trade-off between performance and speed.

2.3.7 Conclusion

In this work we have developed a unified deep architecture that is able to jointly reason about classification, detection and semantic segmentation. Our approach is very simple, it can be trained end-to-end and it performs extremely well in the challenging KITTI dataset, outperforming the state of the art in the road segmentation task. Our approach is also computationally efficient, taking just 42.48 ms to perform all tasks.

While our method can improve inference speed and computational efficiency, the performance improvement in terms of prediction accuracy is negligible compared to the individual tasks. This is despite very favourable conditions for multi-task learning. All three tasks are learned on the same dataset with a lack of segmentation annotation for many images. In addition, the dataset is quite small. So overall we would expect the encoder to be able to effectively utilize the added annotation when training with multiple task, but this is not the case. This indicates that it is not enough to just add additional losses to a network in order to obtain better representations by utilizing more data and annotations.

This is a limitation of simple end-to-end learning. In Chapters 4 and 5 we will show how we can learn better representations using multi-task learning, by utilizing the underlying structures of the tasks and explicitly modelling their relation.

	MaxF1	AP	moderate	hard	m. Acc.	Precision	Recall
VGG-pool5	95.99 %	92.31 %	84.68 %	72.08 %	95.75 %	100 %	91.50 %
ResNet50	96.35 %	92.13 %	86.92 %	72.75 %	98.36 %	100 %	96.73 %
ResNet101	95.99 %	91.99 %	89.30 %	75.42 %	98.61 %	99.33 %	97.38 %

Table 2.7 Results of joint training

	speed [msec]	speed [fps]
VGG-pool5	42.48 ms	23.53 Hz
ResNet50	60.22 ms	16.60 Hz
ResNet101	79.70 ms	12.54 Hz

Table 2.8 Speed of joint inference.

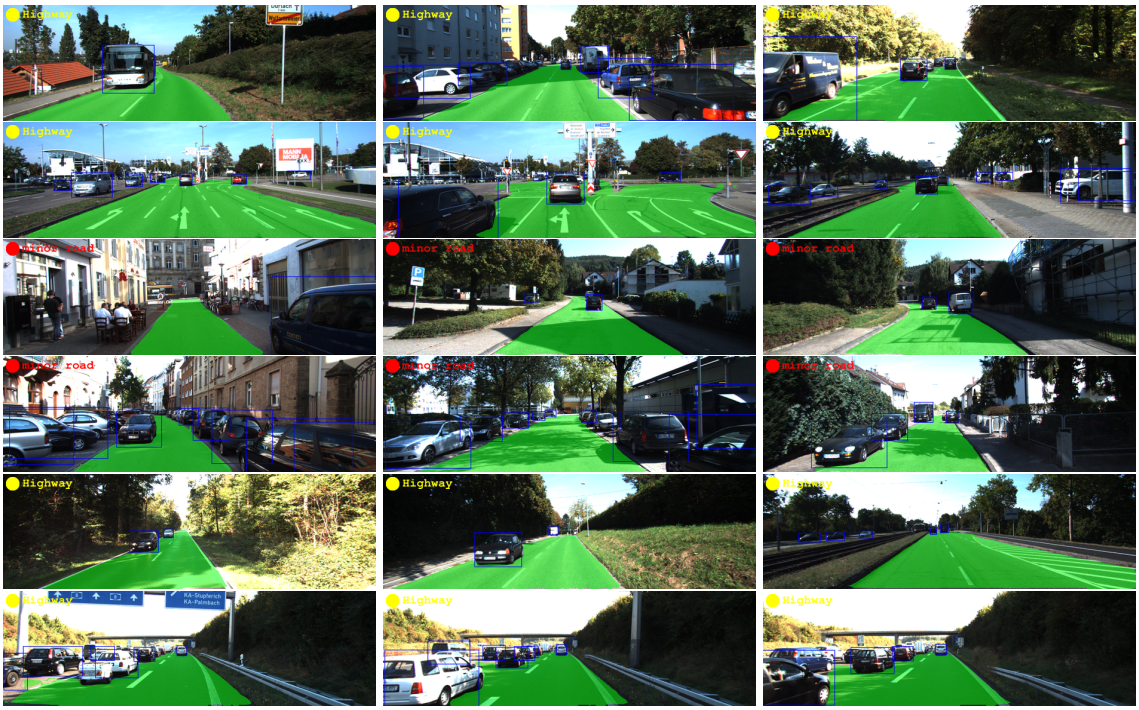


Fig. 2.9 Visualization of the MultiNet output.

Chapter 3

Learning Structured with CRFs

In this chapter we investigate the modelling capabilities of Conditional Random Fields (CRFs). CRFs are able to model complex dependencies when predicting multiple variables. For structured tasks like semantic segmentation, where the goal is to predict many variables, the best performing models have traditionally combined the structured modelling capabilities of CRFs with the feature extraction power of CNNs. In this role CRFs enable the model to learn structure end-to-end, entirely from data, following the classic deep learning paradigm.

In more recent works, however, CRF post-processing has fallen out of favour. We argue that this is mainly due to the slow training and inference speeds of CRFs, as well as the difficulty of learning the internal CRF parameters. To overcome both issues we propose to add the assumption of conditional independence to the framework of fully connected CRFs. This allows us to reformulate the inference in terms of convolutions, which can be implemented highly efficiently on GPUs. Doing so speeds up inference and training by two orders of magnitude. All parameters of the convolutional CRFs can easily be optimized using back-propagation.

This chapter is largely based on my paper *Convolutional CRFs for semantic segmentation* (Teichmann and Cipolla, 2019) which was published at the British Machine Vision Conference (BMVC) 2019 as an oral presentation. The publication has inspired further CRF research and was cited more than 12 times.

3.1 Introduction

Semantic image segmentation, which aims to produce a categorical label for each pixel in an image, is a very import task for visual perception. Convolutional Neural Networks have been proven to be very strong in tackling semantic segmentation tasks (Chen et al., 2018, 2017a; Long et al., 2015; Zhao et al., 2017). While simple feed-forward CNNs are extremely

powerful in extracting local features and performing good predictions utilizing a small field of view, they lack the capability to utilize context information and they cannot model interactions between predictions directly. Thus it has been suggested that such deep neural networks may not be the perfect model for structured predictions tasks such as semantic segmentation (Lin et al., 2016; Zhao et al., 2017; Zheng et al., 2015a). Several authors have successfully combined the effectiveness of CNNs to extract powerful features, with the modelling power of CRFs in order to address the discussed issues (Chandra and Kokkinos, 2016; Lin et al., 2016; Zheng et al., 2015a). Despite their indisputable success, structured models have fallen out of favour in more recent approaches (Chen et al., 2017a; Wu et al., 2016; Zhao et al., 2017).

We believe that the main reasons for this development are that CRFs are notoriously slow and hard to optimize. Learning the features for the structured component of the CRF is an open research problem (Lin et al., 2016; Vemulapalli et al., 2016) and many approaches rely on entirely hand-crafted Gaussian features (Chen et al., 2018; Krähenbühl and Koltun, 2011; Schwing and Urtasun, 2015; Zheng et al., 2015a). In addition, CRF inference is typically two orders of magnitude slower than CNN inference. This makes CRF based approaches too slow for many practical applications. The long training times of the current generation of CRFs also make more in-depth research and experiments with such structured models impractical.

To solve both of these issues we propose to add the strong and valid assumption of conditional independence to the existing framework of fully connected CRFs (FullCRFs) introduced by Krähenbühl and Koltun (2011). This allows us to reformulate a large proportion of the inference as convolutions, which can be implemented highly efficiently on GPUs. We call our method convolutional CRF (ConvCRF). Back-propagation (Rumelhart et al., 1986) can be used to train all parameters of the ConvCRF. Inference in convolutional CRFs (ConvCRFs) can be performed in less than 10ms. This is a speed increase of two orders of magnitude compared to FullCRFs. We believe that those fast train and inference speeds will greatly benefit future research and hope that our results will help to revive CRFs as a popular method to solve structured learning tasks.

3.2 Related Work

Recent advances in semantic segmentation are mainly driven by powerful deep neural network architectures (He et al., 2016a; Krizhevsky et al., 2012a; Simonyan and Zisserman, 2014a; Wu et al., 2016). Following the ideas introduced by Long et al. (2015), transposed convolution layers are applied at the end of the prediction pipeline to produce high resolution output.

Atrous (dilated) convolutions (Chen et al., 2015a; Yu and Koltun, 2015b) are commonly applied to preserve spatial information in feature space.

Many architectures have been proposed (Badrinarayanan et al., 2017; Noh et al., 2015; Paszke et al., 2016; Ronneberger et al., 2015b; Teichmann et al., 2018; Wu et al., 2016), based on the ideas above. All of those approaches have in common that they primarily rely on the powerful feature extraction provided by CNNs. Predictions are pixel-wise and conditionally independent (given the common feature base of nearby pixels). Structured knowledge and background context is ignored in these models.

One popular way to integrate structured predictions into CNN pipelines is to apply a fully connected CRF (FullCRF) (Krähenbühl and Koltun, 2011) on top of the CNN prediction (Chandra and Kokkinos, 2016; Chen et al., 2015a; Lin et al., 2016; Schwing and Urtasun, 2015; Zheng et al., 2015a). Utilizing the edge-awareness of CRFs, FullCRFs have been successfully utilized to solve weakly- and semi-supervised segmentation tasks (He and Zemel, 2009; Li et al., 2018a; Tang et al., 2018; Triggs and Verbeek, 2008). Tang et al. (2018) propose to use a CRF based loss function. All of those approaches can benefit from our contributions.

Parameter learning in CRFs FullCRFs rely on hand-crafted features for the pairwise (Gaussian) kernels. In their first publication Krähenbühl and Koltun (2011) optimized the remaining parameters with a combination of expectation maximization and grid-search. In a follow-up work Krähenbühl and Koltun (2013) proposed to use gradient descent. Their idea utilizes the fact that all operations are linear with respect to the Gaussian kernel k_G. This allows them to optimize all internal CRF parameters, using gradient descend, without requiring gradients with respect to k_G. However such an approach comes with limitations. The Gaussian kernel k_G itself cannot be learned using such an approach. CRFasRNN (Zheng et al., 2015a) uses the same ideas to implement joint CRF and CNN training. This however still requires hand-crafted pairwise (Gaussian) features.

Quadratic optimization (Chandra and Kokkinos, 2016; Vemulapalli et al., 2016) has been proposed to learn the Gaussian features. These approaches however do not fit well into many deep learning pipelines. Another way of learning the pairwise features is piecewise training (Lin et al., 2016). An additional advantage of this method is that it avoids repeated CRF inference, speeding up the training considerably. However, this approach is of an approximate nature, and inference speed is still very slow.

Inference speed of CRFs. In order to circumvent the issue of very long training and inference times, some CRF based pipelines produce an output which is down-sampled by a

factor of 8×8 (Chandra and Kokkinos, 2016; Lin et al., 2016). This speeds up the inference considerably. However, this harms their predictive capabilities. Deep learning based semantic segmentation pipelines perform best when they are challenged to produce a full-resolution prediction (Chen et al., 2017a; Long et al., 2015; Yu and Koltun, 2015b). To the best of our knowledge, no significant progress in inference speed has been made since the introduction of FullCRFs (Krähenbühl and Koltun, 2011).

3.3 Fully Connected CRFs

In the context of semantic segmentation most recent CRF based approaches are based on the FullCRF model introduced by Krähenbühl and Koltun (2011). Consider an input image I consisting of n pixels and a segmentation task with k classes. A segmentation of I is then modelled as a random field $\mathbf{X} = \{X_1, \ldots, X_n\}$, where each random variable X_i takes values in $\{1, \ldots, k\}$, i.e. the label of pixel i. Solving $\arg\max_L P(X = L|I)$ then leads to a segmentation L of the input image I. $P(X|I)$ is modeled as a CRF over the Gibbs distribution:

$$P(X = \hat{x}|\tilde{I} = I) = \frac{1}{Z(I)} exp(-E(\hat{x}|I)) \tag{3.1}$$

where the energy function $E(\hat{x}|I)$ is given by

$$E(\hat{x}|I) = \sum_{i \leq n} \psi_u(\hat{x}_i|I) + \sum_{i \neq j \leq N} \psi_p(\hat{x}_i, \hat{x}_j|I). \tag{3.2}$$

The function $\psi_u(x_i|I)$ is called unary potential. The unary itself can be considered a segmentation of the image, and any segmentation pipeline can be used to predict the unary. In practice, most newer approaches (Chen et al., 2018; Schwing and Urtasun, 2015; Zheng et al., 2015a) use CNNs to compute the unary.

The function $\psi_p(x_i, x_j|I)$ is the pairwise potential. It accounts for the joint distribution of pixels i, j. It allows us to explicitly model interactions between pixels, e.g. pixels with similar colour are likely the same class. In FullCRFs ψ_p is defined as weighted sum of Gaussian kernels $\mathbf{k}_g^{(1)} \ldots \mathbf{k}_g^{(m)}$:

$$\psi_p(x_i, x_j|I) := \mu(x_i, x_j) \sum_{m=1}^{M} w^{(m)} \mathbf{k}_g^{(m)}(\mathbf{f}_i^I, \mathbf{f}_j^I), \tag{3.3}$$

where $w^{(m)}$ are learnable parameters. The feature vectors $\mathbf{f}_i^I$ can be chosen arbitrarily and may depend on the input Image I. The function $\mu(x_i, x_j)$ is the compatibility transformation, which only depends on the labels x_i and x_j, but not on the image I.

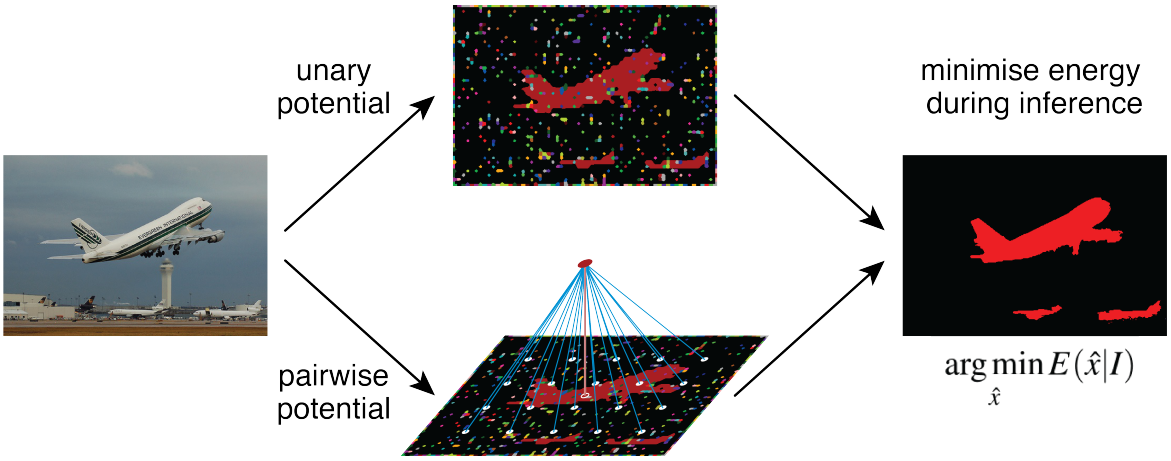


Fig. 3.1 Visualization of CRF inference. The constraints of the unary and pairwise potential are balanced by minimizing the energy function during inference. Image courtesy of Kristina Klein.

A very widely used compatibility function (Chen et al., 2018; Krähenbühl and Koltun, 2011; Zheng et al., 2015a) is the Potts model $\mu(x_i, x_j) = 1_{[xi \neq xj]}$. This model tries to assign pixels that have similar features the same prediction. Zheng et al. (2015a) propose to use 1×1 convolutions as the compatibility transformation. Such a function allows the model to learn more structured interactions between predictions.

FullCRFs utilize two Gaussian kernels with hand-crafted features. The appearance kernel uses the raw colour values I_j and I_i as features. The smoothness kernel is based on the spatial coordinates p_i and p_j. The entire pairwise potential is then given as:

$$\mathbf{k}(\mathbf{f}_i^I, \mathbf{f}_j^I) := w^{(1)} exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2}\right) + w^{(2)} exp\left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2}\right), \tag{3.4}$$

where $w^{(1)}$, $w^{(2)}$, as well as θ_α, θ_β and θ_γ are the only learnable parameters of the model. Most CRF based segmentation approaches (Chen et al., 2018; Schwing and Urtasun, 2015; Zheng et al., 2015a) use the very same hand-crafted pairwise potentials proposed by Krähenbühl and Koltun (2011). CRFs are notoriously hard to optimize, and utilizing hand-crafted features circumvents this problem. We visualize inference in CRFs in Figure 3.1.

3.3.1 Mean Field Inference

Inference in CRFs is performed by estimating the distribution $P(X)$. Given $P(X)$ a prediction can then be obtained by computing $\arg\max_L P(X = L|I)$. However finding an exact solution

for $P(X)$ is NP-complete (Koller and Friedman, 2009, pp. 288 - 290) hence intractable. Krähenbühl and Koltun (2011) therefore propose to use the mean field approximation (algorithm 1) for inference in FullCRFs.

The mean field algorithm computes a distribution $Q(X)$ which approximates $P(X)$. The mean field algorithm does provide some strong theoretical guaranties (Koller and Friedman, 2009, pp. 454 - 456). The mean field algorithm yields a distribution Q which can be expressed as a product of independent marginals $Q(X) = \prod_i Q_i(X_i)$. Furthermore among all distributions Q^* with that property the mean field algorithm is guarantied to converge towards the distributions Q which minimizes the KL-divergence $D(Q||P)$.

All steps of the mean field Algorithm (algorithm 1) other than the message passing, are highly parallelized and can be implemented easily and efficiently on GPUs using standard deep learning libraries. Zheng et al. (2015a) therefore propose to integrate the mean field algorithm into a CNN by unrolling the outer loop and computing exactly 5 iterations. The entire model can then be trained using back-propagation to minimize a suitable loss function like softmax cross-entropy.

Algorithm 1 Mean field approximation in fully connected CRFs

Input: Image I, Unary $\psi_u(x_i|I)$ and pairwise $\mathbf{k}_g^{(m)}(\mathbf{f}_i^I, \mathbf{f}_j^I)$ potentials
Output: Q, where $Q_i(l)$ approximates the likelihood that label l is at position i, i.e. $P(x_i = l)$

1: Initialize: ▷ $Q_i \leftarrow \frac{1}{Z_i} exp(-\psi_u(x_i|I))$ "softmax"
2: **while** not converged **do**
3: $\tilde{Q}_i(l) \leftarrow \sum_{i \neq j} w^{(m)} \mathbf{k}_g^{(m)}(\mathbf{f}_i^I, \mathbf{f}_j^I) Q_i(l)$ ▷ Message Passing
4: $\hat{Q}_i(l) \leftarrow \sum_{l' \in L} \mu(l, l') \tilde{Q}_i(l')$ ▷ Compatibility Transformation
5: $\check{Q}_i(l) \leftarrow \psi_u(x_i|I) + \hat{Q}_i(l)$ ▷ Adding Unary Potentials
6: $Q_i(l) \leftarrow \text{normalize}(\check{Q}_i(l))$ ▷ e.g. softmax
7: **end while**

The message passing however is the bottleneck of the CRF computation. Exact computation is quadratic in the number of pixels and therefore infeasible. Krähenbühl and Koltun (2011) instead proposed to utilize the permutohedral lattice (Adams et al., 2010) approximation, a high-dimensional filtering algorithm. The permutohedral lattice however is based on a complex data structure. While there is a very sophisticated and fast CPU implementation, the permutohedral lattice does not follow the SIMD (Nickolls et al., 2008) paradigm of efficient GPU computation. In addition, efficient gradient computation of the permutohedral lattice approximation is also a non-trivial problem. This is the underlying reason why FullCRF based approaches use hand-crafted features.

3.4 Convolutional CRFs

The convolutional CRFs (ConvCRFs) supplement FullCRFs with a conditional independence assumption. We assume that the label distribution of two pixels i, j is conditionally independent, if for the Manhattan distance d holds $d(i, j) > k$, where k is called filter size.

We argue that this assumption is strong and valid. It is valid because it is based on the same locality assumption used by CNNs, which are used very succesfully in the image domain. It is a strong assumption since it implies that the pairwise potential is zero, for all pixels whose distance exceeds k. This reduces the complexity of the pairwise potential from quadratic to linear. Overall this produces a very promising theoretical foundation for our ConvCRFs model. Strong and valid assumptions are the powerhouse of machine learning modelling. Utilizing good assumption is the main way to improve a statistical model given limited amount of data (Blumer et al., 1989; Wolpert et al., 1997).

3.4.1 Efficient Message Passing in ConvCRFs

One of the key contributions of the work presented in this chapter is to show that exact message passing is efficient in the ConvCRFs model. This eliminates the need to use the permutohedral lattice approximation, making both highly efficient GPU computation and complete feature learning possible. Towards this goal we reformulate the message passing step to be a convolution with truncated Gaussian kernel and observe that this can be implemented in a very similar way to regular convolutions in CNNs.

Consider an input $\mathbf{P}$ with shape $[bs, c, h, w]$ where bs, c, h, w denote batch size, number of classes, input height and width respectively. For a Gaussian kernel g, defined by feature vectors $\mathbf{f}_1 \dots \mathbf{f}_d$, each of shape $[bs, h, w]$, we define its kernel matrix by

$$\mathbf{k}_g[b, dx, dy, x, y] := exp\left(-\sum_{i=1}^{d} \frac{|\mathbf{f}_i[b, x, y] - \mathbf{f}_i[b, x - dx, y - dy]|^2}{2\theta_i^2}\right), \tag{3.5}$$

where θ_i is a learnable parameter. For a set of Gaussian kernels $g_1 \dots g_s$ we define the merged kernel matrix $\mathbf{K}$ as $\mathbf{K} := \sum_{i=1}^{s} w_i \cdot g_i$. The result Q of the combined message passing of all s kernels is now given as:

$$Q[b, c, x, y] = \sum_{dx, dy \leq k} \mathbf{K}[b, dx, dy, x, y] \cdot \mathbf{P}[b, c, x + dx, y + dy]. \tag{3.6}$$

This message passing operation is similar to standard 2d-convolutions of CNNs. In our case however, the filter values depend on the spatial dimensions x and y. This is similar to locally connected layers (Chen et al., 2015c). However, unlike locally connected layers (and

unlike 2d-convolutions), our filters are constant in the channel dimension c. One can view our operation as convolution over the dimension c.

It is possible to implement our convolution operation by using standard CNN operations only. This however requires the data to be reorganized in GPU memory several times, which is a very slow process. Profiling shows that 90% of GPU time is spent for the reorganization of data. We therefore opted to build a native low-level implementation, to gain an additional 10-fold speed-up.

Efficient computation of our convolution can be implemented analogously to 2d-convolution (and locally connected layers). The first step is to tile the input P in order to obtain data with shape $[bs, c, k, k, h, w]$. This process is usually referred to as *im2col* and the same as in 2d-convolutions (Chetlur et al., 2014); 2d-convolutions proceed by applying a batched matrix multiplication over the spatial dimension. We replace this step with a batched dot-product over the channel dimension. All other steps are the same as in 2d-convolutions.

This allows us to implement our message passing using the same CUDA kernels which are also used for CNN convolution computation. Those CUDA kernels which are implemented in Nvidias cudnn library (Chetlur et al., 2014) are highly optimized, both in software but also in hardware. Current GPUs are designed to perform very well on those CUDA kernels, part of the computation is implemented in hardware on tensor cores whose specific purpose it is to speed up CNN computation. This is the main reason for the speed improvement of our approach compared to FullCRFs. FullCRFs use the permutohedral lattice approximation for message passing, which is by far its largest computational bottleneck. Indeed all computation steps apart from message passing are the same in FullCRFs and ConvCRFs, so all the speed improvement we report comes from our very efficient message passing implementation. In theory our message passing has the same linear time complexity as the permutohedral lattice approximation. However in practice the permutohedral lattice approximation is based on a complex data structure, designed for CPU computation while we are able to benefit from Nvidias hard- and software optimizations.

3.4.2 Additional Implementation Details

For the sake of comparability we use the same design choices as FullCRFs in our baseline ConvCRF implementation. In particular, we use softmax normalization, the Potts model and the same hand-crafted Gaussian features as proposed by Krähenbühl and Koltun (2011). Analogous to Krähenbühl and Koltun (2011) we also apply Gaussian blur to the pairwise kernels. This leads to an increase of the effective filter size by a factor of 4.

As proposed by Zheng et al. (2015b) we integrate the CRF computation inside our CNN model by unrolling the outer loop of the mean-field algorithm. Like Zheng et al. (2015b) we

compute exactly 5 iterations of the algorithm. We train the entire model by minimizing the softmax cross-entropy loss between prediction and target.

In additional experiments we investigate the capability of our CRFs to learn Gaussian features. Towards this goal we replace the input features p_i of the smoothness kernel with learnable variables. Those variables are initialized to the same values as the hand-crafted version, but are adjusted as part of the training process. We also implement a learnable compatibility transformation using a 1×1 convolution, following the ideas of (Zheng et al., 2015a).

3.5 Experimental Evaluation

Dataset: We evaluate our method on the challenging Pascal VOC 2012 (Everingham et al., 2012) image dataset. Following the literature (Chen et al., 2018; Long et al., 2015; Wu et al., 2016; Zhao et al., 2017) we use the additional annotation provided by Hariharan et al. (2011) resulting in 10.582 labelled images for training. Out of those images we hold back 200 images to fine tune the internal CRF parameters and use the remaining 10.382 to train the unary CNN. We report our results on the 1464 images of the official validation set.

Unary: We train a ResNet101 (He et al., 2016a) to compute the unary potentials. We use the ResNet101 implementation provided by the PyTorch (Paszke et al., 2017) repository. A simple FCN (Long et al., 2015) is added on top of the ResNet to decode the CNN features and obtain valid segmentation predictions. The network is initialized using ImageNet Classification weights (Deng et al., 2009b) and then trained on Pascal VOC data directly. Unlike many other projects, we do not utilize larger segmentation datasets such as MS COCO (Lin et al., 2014) for pre-training, we only use the images provided by the Pascal VOC 2012 benchmark.

The CNN is trained for 200 epochs using a batch size of 16 and the adam optimizer (Kingma and Ba, 2014b). The initial learning rate is set to 5×10^{-5} and polynomially decreased (Chen et al., 2018; Liu et al., 2015b) by multiplying the initial learning rate with $((1 - \frac{step}{max\_steps})^{0.9})^2$. An L_2 weight decay with factor 5×10^{-4} is applied to all kernel weights, and 2d-Dropout (Tompson et al., 2015) with rate 0.5 is used on top of the final convolutional layer. The same hyper-parameters are also used for the end-to-end training.

The following data augmentation methods are applied: random horizontal flip, random rotation ($\pm 10°$) and random resize with a factor in (0.5, 2). In addition, the image colours are jittered using random brightness, random contrast, random saturation and random hue.

(a) Image (b) Label (c) Noised (d) ConvCRF11 (e) FullCRF

Fig. 3.2 Visualization of the synthetic task. Especially in the last example, the artifacts from the permutohedral lattice approximation can clearly be seen at object boundaries.

All random numbers are generated using a truncated normal distribution. The trained model achieves validation mIoU of 71.23 % and a train mIoU of 91.84 %.

CRF: Following the literature (Chen et al., 2018; Krähenbühl and Koltun, 2011; Lin et al., 2016; Zheng et al., 2015a), the mean-field inference of the CRF is computed for 5 iterations in all experiments.

3.5.1 ConvCRF on Synthetic Data

To show the capabilities of convolutional CRFs we first evaluate their performance on a synthetic task. We use the Pascal VOC (Everingham et al., 2012) dataset as a basis, but augment the ground truth towards the goal to simulate prediction errors. The noised labels are used as unary potentials for the CRF, and the CRF is then challenged to denoise the predictions. Finally, the output of the CRF is compared to the original label of the Pascal VOC dataset.

Method	Unary	FullCRF	Conv5	Conv7	Conv11	Conv13	Conv15
Speed [ms]	68	647	7	13	26	34	45
Accuracy [%]	86.60	94.79	97.13	97.13	98.97	98.99	98.95
mIoU [%]	51.87	84.37	90.90	92.98	93.74	93.89	93.71

Table 3.1 Performance comparison of CRFs on the synthetic benchmark. The speed tests have been done on a Nvidia GeForce GTX 1080 Ti GPU and an Intel Xeon E5-2630 CPU. The images are processed in full resolution. ConvCRF use GPU computation while FullCRF inference is computed on a CPU. Conv7 denotes a ConvCRF with filter size 7.

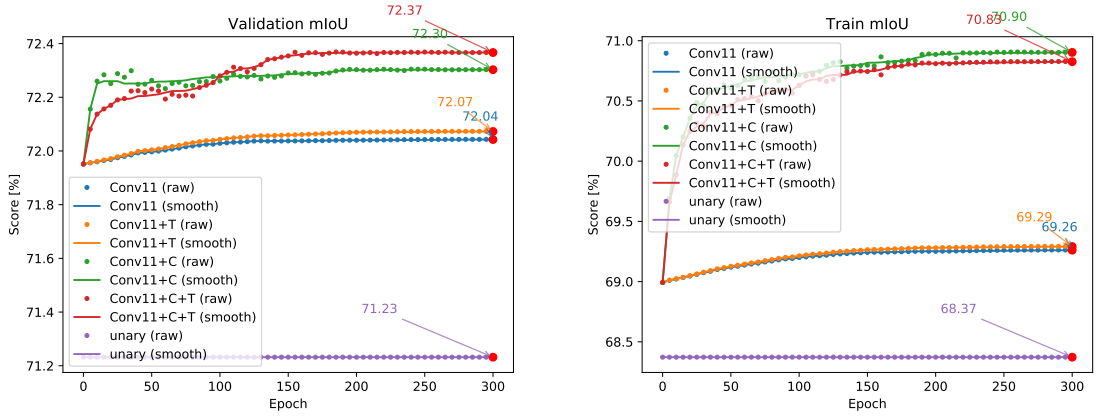
Towards the goal of creating a relevant task, the following augmentation procedure is used: first the ground truth is down-sampled by a factor of 8. Then random noise is added to the predictions and the result is up-sampled to the original resolution again. This process simulates inaccuracies as a result of the low resolution feature processing of CNNs, as well as prediction errors similar to the checkerboard issue found in deconvolution based segmentation networks (Gao et al., 2017; Odena et al., 2016). Some examples of the augmented ground truth are shown in Figure 3.2.

In our first experiment we compare FullCRFs and ConvCRFs using the exact same parameters. To do this we utilize the hand-crafted Gaussian features. The remaining five parameters ($w^{(1)}$, $w^{(2)}$, as well as θ_α, θ_β and θ_γ) are initialized to the default values proposed by Krähenbühl and Koltun (2011). Note that this gives FullCRFs a natural advantage, since those values were chosen such that FullCRFs perform well. The performance of CRFs however is very robust with respect to these five parameters (Krähenbühl and Koltun, 2011).

The results of our first experiment are given in Table 3.1. It can be seen that ConvCRFs outperform FullCRFs significantly. This shows that ConvCRFs are structurally superior to FullCRFs. The better performance of ConvCRFs with the same parameters can be explained by our exact message passing, which avoids the approximation errors compared to the permutohedral lattice approximation. We provide a visual comparison in Figure 3.2, where ConvCRF clearly provides higher quality output. The FullCRF output shows approximation artefacts at the boundary of objects. In addition, we note that ConvCRFs are faster by two orders of magnitude, making them favourable in almost every use case.

3.5.2 Decoupled Training of ConvCRF

In this section we discuss our experiments on Pascal VOC data using a two-stage training strategy. First the unary CNN model is trained to perform semantic segmentation on the Pascal VOC data. Those parameters are then fixed, and in the second stage the internal CRF



(a) Validation mIoU over time

(b) Train mIoU over time

Fig. 3.3 Training and validation mIoU over time for decoupled training. Approaches marked with +C use convolutions as compatibility transformation and +T shows learning of Gaussian features.

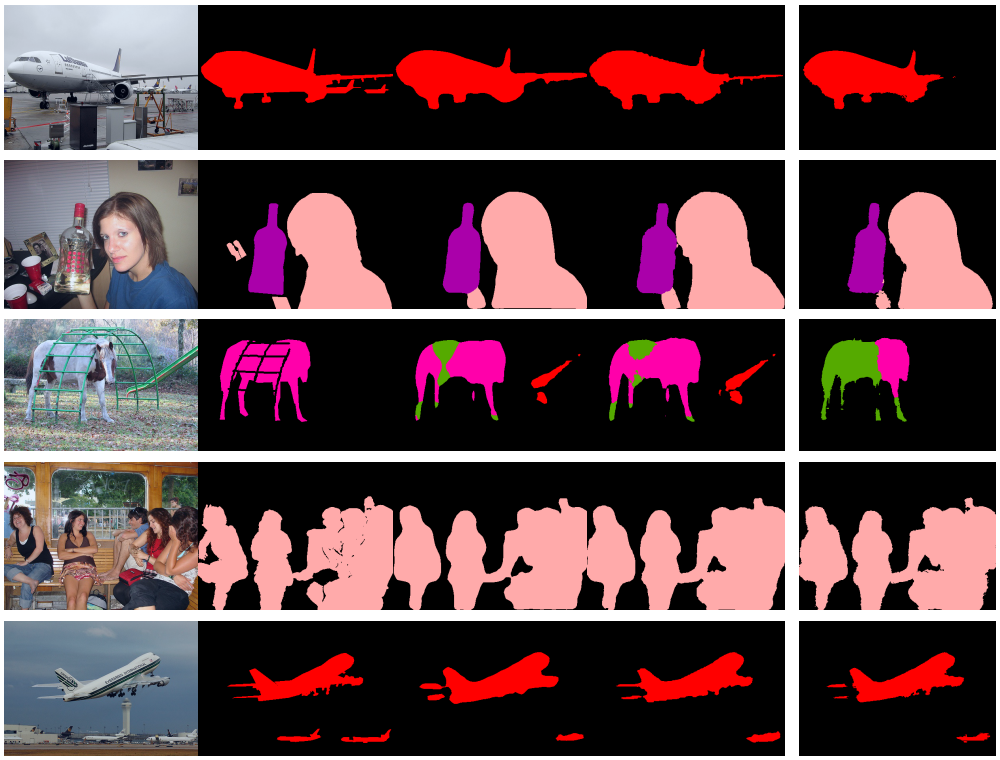
Method	Unary	DeepLab	ConvCRF	Conv+T	Conv+C	**Conv+CT**
mIoU [%]	71.23	72.02	72.04	72.07	72.30	**72.37**
Accuracy [%]	93.80	94.01	93.99	94.01	94.01	**94.03**
train_crf mIoU [%]	68.37	68.61	69.26	69.29	70.90	**70.83**

Table 3.2 Performance comparison of CRFs on validation data using decoupled training. +C uses convolutions as compatibility transformation and +T learns the Gaussian features. The same unaries were used for all approaches; only the CRF code from DeepLab was utilized. Train_crf mIoU denotes the mIoU computed on the set used for training the CRF consisting of 200 images.

parameters are optimized with respect to the CNN predictions. The same unary predictions are used across all experiments, to reduce variants between runs.

Decoupled training has various merits compared to an end-to-end pipeline. Firstly it is very flexible. A standalone CRF training can be applied on top of any segmentation approach. The unary predictions are treated as a black-box input for the CRF training. In practice this means that the two training stages do not need to interface at all, making fast prototyping very easy. Additionally decoupled training keeps the system interpretable. Lastly, piecewise training tackles the vanishing gradient problem (Bengio et al., 1994), which is still an issue in CNN based segmentation approaches (Wu et al., 2016). This leads overall to faster, more robust and reliable training.

For our experiments we train the CRF models on the 200 held-out images from the training set and evaluate the CRF performance on the 1464 images of the official Pascal



(a) Image (b) Label (c) Unary (d) ConvCRF (e) DeepLab-CRF

Fig. 3.4 Visualization of results on Pascal VOC data using a decoupled training strategy. Examples 2 and 4 depict failure cases, in which the CRFs are not able to improve the unary.

VOC dataset. We compare the performance of the ConvCRF with filter size 11 to the unary baseline results as well as a FullCRF trained following the methodology of DeepLab (Chen et al., 2018).

We report our results in Table 3.2, and the training curves are visualized in Figure 3.3. In all experiments, applying CRFs boosts the performance considerably. The experiments also confirm the observation of Section 3.5.1, that ConvCRF perform slightly better than FullCRFs. We also observe that the ConvCRF implementation utilizing a learnable compatibility transformation as well as learnable Gaussian features performs best. Model output is visualized in Figure 3.4.

3.5.3 End-to-End Learning with ConvCRFs

In this section we discuss our experiments using an end-to-end learning strategy for ConvCRFs. In end-to-end training, the gradients are propagated through the entire pipeline. This allows the CNN and CRF model to co-adapt and therefore to produce the optimum output w.r.t the entire network. The down-side of end-to-end training is that the gradients need

Method	Unary	CRFasRNN	ConvCRF
mIoU [%]	71.23	72.07	72.27
Accuracy [%]	93.80	94.07	94.11
train mIoU [%]	91.84	93.70	92.31

Table 3.3 Performance comparison of end-to-end trained CRFs.

to be propagated through five iterations of the mean-field inference, resulting in vanishing gradients (Zheng et al., 2015a).

We train our network for 300 epochs using a training protocol similar to CRFasRNN (Zheng et al., 2015a). We first train the unary potential until convergence, following the training procedure of the decoupled training. In a second step we optimize the CRF and CNN jointly using a greatly reduced learning rate of 10^{-8}. We use a batch-size of 8 for the joint optimization. In this regard we differ from (Zheng et al., 2015a), who propose to reduce the batch size to 1 for the joint training.

The entire training process takes about 30 hours using four 1080Ti GPUs in parallel. We believe that the fast training and inference speeds will greatly benefit and ease future research using CRFs. We compare our ConvCRF to the approach proposed in CRFasRNN (Zheng et al., 2015a) and report the results in Table 3.3. Overall we see that ConvCRF slightly outperforms CRFasRNN at a much higher speed.

3.6 Conclusion

In this chapter we proposed convolutional CRFs, a novel CRF design. Adding the strong and valid assumption of conditional independence enables us to remove the permutohedral lattice approximation. This allows us to implement the message passing highly efficiently on GPUs as convolution operations. This increases training and inference speed by two orders of magnitude. In addition, we observe a modest accuracy improvement when computing the message passing exactly. Our method also enables us to easily train the Gaussian features of the CRF using back-propagation.

Limitations & further work: We designed and implemented our ConvCRF model to be as similar as possible to FullCRFs. Our goal was to provide a fast but simple plug-and-play alternative to the still widely used CRF implementation. Our experiments in Section 3.5.1 indicate that we have succeeded in this. We are able to run ConvCRFs using internal parameters optimized for FullCRFs. However, this limits the ability of our model to learn

more complicated structure from data and it leaves a lot of room for further research. In further work, we would like to explore the option to learn more internal CRF parameters and experiment with more advanced CRF architectures. We would also like to use the modelling capabilities of CRFs to solve additional tasks such as instance segmentation, a task that is particularly suitable for CRFs since the entire goal of this task is to predict the relation between two pixels.

Chapter 4

Improving Generalization, Interpretability and Robustness using Structured Modelling

In the last chapter we discussed how we can learn structure as part of our network. This follows the classic deep learning paradigm to learn all aspects of the model from data. In the next two chapters we will look at explicitly modelling structure towards the goal of utilizing our prior knowledge about the nature of the task. Towards this goal we build a model that computes an embedding, an intermediate representation that can be used to solve the actual task, utilizing the underlying geometric structure.

In this chapter we present our localization work which solves the task of predicting the camera pose of scenes which have been previously visited. Towards this goal we build a system that estimates 3D geometry and simultaneously recognizes surrounding objects. Using the 3D geometry as an intermediate representation we can solve the pose estimation as a perspective-n-point problem (PnP) (Kneip et al., 2014; Lepetit et al., 2009) and compute the exact camera pose very robustly by utilizing Random sample consensus (RANSAC).

This chapter is largely based on my paper *Large Scale Joint Semantic Re-localisation and Scene Understanding via Globally Unique Instance Coordinate Regression* Budvytis*, Teichmann*, Vojir* and Cipolla (2019) which was published at the British Machine Vision Conference (BMVC) 2019 and is a joint work with Ignas Budvytis and Tomas Vojir.

*This authors are equally contributing first authors

4.1 Overview

In this chapter we present a novel approach to joint semantic localization and scene understanding. Our work is motivated by the need for localization algorithms which not only predict 6-DoF camera pose but also simultaneously recognize surrounding objects and estimate 3D geometry. Such capabilities are crucial for computer vision guided systems that interact with the environment: autonomous driving, augmented reality and robotics. In particular, we propose a two-step procedure. During the first step we train a convolutional neural network to jointly predict per-pixel globally unique instance labels (Budvytis et al., 2018) and corresponding local coordinates for each instance of a static object (e.g. a building). During the second step, we obtain scene coordinates (Shotton et al., 2013) by combining object center coordinates and local coordinates and use them to perform 6-DoF camera pose estimation. We evaluate our approach on real world (CamVid-360) and artificial (SceneCity) autonomous driving datasets (Budvytis et al., 2018).

As computer vision enabled robotic systems are increasingly deployed in the real world, simplicity, efficiency, verifiability and robustness of computer vision algorithms become highly important aspects of their design. An example of a desired solution satisfying the aforementioned requirements would likely include training a single network that would predict a structured semantically meaningful output, the correctness of which could be verified at test time and from which all the necessary tasks for navigation and interaction with the environment could be performed. For example, employing multiple different networks for object detection, segmentation and localization can quickly exhaust available computing resources and adds unnecessary complexity and integration issues.

In this chapter we propose such a structured representation from which tasks of semantic segmentation, recognition and localization can be performed efficiently and accurately. Our proposed solution is inspired by works on globally unique instance segmentation (Budvytis et al., 2018) and scene coordinate regression (Brachmann et al., 2017; Brachmann and Rother, 2018; Shotton et al., 2013). It includes the following key steps. First, a dataset of densely sampled images – ideally a video – of the environment is created. It is labelled with globally unique instance labels (Budvytis et al., 2018), and a corresponding 3D point cloud is obtained by running a structure-from-motion algorithm (map, 2019) on the collected images. Second, a CNN is trained to simultaneously predict globally unique instance labels and local coordinates of corresponding objects. Finally, at test time, scene coordinates are formed by combining object centre coordinates with local coordinates for a 6-DoF camera pose estimation which is formulated as a solution to a PnP (Kneip et al., 2014; Lepetit et al., 2009). See Figure 4.1 for an example output of our method.

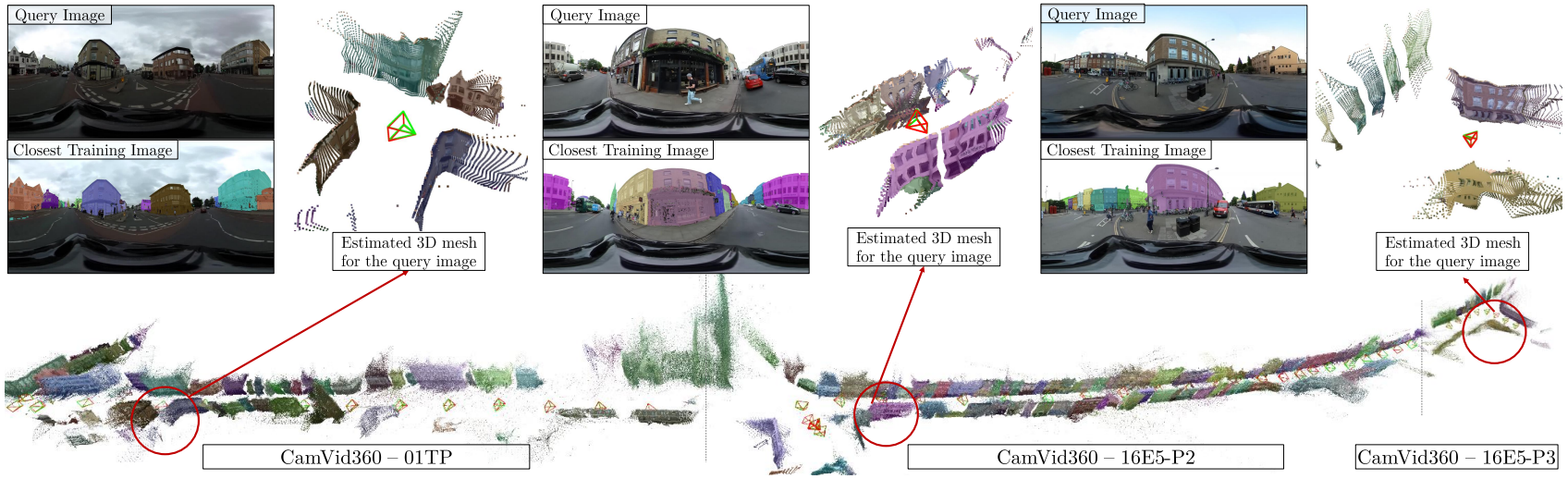


Fig. 4.1 Three triplets of images on the top of this figure illustrate a typical result of our framework for joint semantic re-localization and scene understanding via globally unique instance coordinate prediction. The top left image of the triplet corresponds to a query image provided as an input to our network. The output of the network consists of per-pixel 3D coordinates, and corresponding globally unique instance labels are shown on the right together with ground truth (green) and estimated (red) camera poses. The image at the bottom left shows the closest image in the database, with ground truth building instance label images overlaid. The bottom part of the figure illustrates a cumulative 3D point cloud built from predicted scene coordinates as well as ground truth (green) and estimated (red) camera poses for sequences 16E5-P2, 16E5-P3 and 01TP. See Section 4.4 for more quantitative and qualitative results.

Our work differs from classical scene coordinate regression works (Brachmann et al., 2017; Brachmann and Rother, 2018) in two important ways. First, we predict not only scene coordinates but also corresponding globally unique instance labels. Second, we separate scene coordinates into local coordinates and object centre coordinates. Finally, we demonstrate that localization and its output verification is possible from approximate 3D maps where buildings are represented as 3D cuboids as opposed to computing-heavy 3D point clouds. Note that the first extension provides the ability to verify our localization output as in (Budvytis et al., 2018) and the second extension allows for a rapid and accurate 3D point regression enabling scene coordinate prediction of maps larger than 800 buildings and utilize distant objects. The final extension is important for realistic practical localization scenarios where on-vehicle storage of large city maps is not practically desirable.

We evaluate our approach on real world and artificial autonomous driving datasets. Our method predicts more than 53% and 39% of pixels within 50cm of ground truth location for CamVid-360 (Budvytis et al., 2018) and SceneCity Medium (Budvytis et al., 2018) datasets spanning approximately 1.5km and 11.5km in driving length. We obtain 22cm and 20cm median distance error as well as 0.71° and 0.76° median angular errors on estimated camera poses for the same datasets. Our method outperforms competing deep learning based localization methods based on either direct 6-DoF pose prediction (Kendall and Cipolla, 2017; Kendall et al., 2015) or pose estimation from scene coordinates (Brachmann and Rother, 2018) on all datasets. When tested on highly challenging scenarios using a different camera (Google StreetView images) or re-localizing in scenes with missing buildings (Budvytis et al., 2018), our method demonstrates higher robustness than alternative approaches. Our contributions include: (i) a novel formulation of scene coordinate regression as two separate tasks of object instance recognition and local coordinate regression and a demonstration that our proposed solution allows to predict accurate 3D geometry of static objects and estimate 6-DoF pose of camera, (ii) maps larger by several orders of magnitude than previously attempted (Brachmann et al., 2017; Brachmann and Rother, 2018; Li et al., 2018b; Shotton et al., 2013), and (iii) lightweight, approximate 3D maps built from 3D primitives.

The rest of this chapter is structured as follows. Section 4.2 discusses relevant work in localization. Section 4.3 provides details of our proposed localization approach. Section 4.4 describes the experimental setup and corresponding results.

4.2 Related Work

In this section we provide a discussion of various related work on localization.

Matching based localization. Traditionally, large-scale localization problems are formulated as correspondence problems in the image domain or in the 3D point cloud domain. The first group of approaches work by identifying the most similar looking image in a database primarily in two ways: by employing either (i) a pipeline of keypoint detection and matching (Lowe, 2004; Noh et al., 2017b; Yi et al., 2016) or (ii) fast-to-compare image level encoding (Arandjelović et al., 2016; Torii et al., 2015). In order to obtain a 6-DoF pose estimation they are augmented with an additional step of establishing feature matches for one or more neighbour images and solving a perspective-n-point problem (Kneip et al., 2014) inside a RANSAC (Fischler and Bolles, 1981b) solver. The second group of approaches obtain local 3D geometry by using 3D sensors such as structured light (Izadi et al., 2011; Scharstein and Szeliski, 2003), time-of-flight (Wolcott and Eustice, 2014) cameras and RGB based structure from motion (Torii et al., 2009) and match it to a pre-built 3D model of the environment. While both types of work have a potential for providing high accuracy pose estimates at large scale, they are limited by large storage requirements of feature indices or 3D point clouds and relatively slow correspondence estimation procedures. Methods of the latter type are significantly faster, yet as suggested in (Noh et al., 2017b) are more likely to be sensitive to large occlusions and scene changes. Both types of approach provide a location for the whole scene and not of individual objects.

Direct location prediction. The need for test-time storage and correspondence estimation is addressed by the works which attempt to directly predict either a coarse (Weyand et al., 2016) location or a full 6-DoF camera pose (Kendall and Cipolla, 2017; Kendall et al., 2015). An estimation of location or a precise camera pose is obtained by simply training a deep network with a corresponding objective. The coarse methods (Weyand et al., 2016) still require performing additional local feature matching if a 6-DoF pose estimate is needed and hence are not very efficient at test-time. In contrast, methods that directly predict camera pose demonstrate test-time efficiency because only a single pass through a network is required. However, they are prone to over-fitting to the training images (e.g. a network may learn to predict a location based on the presence of a parked car in the image) and are not robust to changes in the environment as shown in Section 4.4 and discussed in detail in (Sattler et al., 2019b).

Localization via scene coordinate prediction. Test time robustness is increased by approaches which perform localization via scene coordinate regression (Brachmann et al., 2017; Brachmann and Rother, 2018; Li et al., 2018b; Shotton et al., 2013). Such works often train a per-pixel 3D scene coordinate regressor, ether by using a CNN (Brachmann et al., 2017) or

other methods (e.g. Random Forest (Shotton et al., 2013)) and solves a perspective-n-point problem to obtain the estimate of the camera pose. Early works focus on learning outlier masks (Shotton et al., 2013), in order to remove unreliable candidates for pose estimation and to propose a differential pose estimation (Brachmann et al., 2017) to be compatible with fully end-to-end training schemes at the expense of a more complex learning task. In contrast, (Brachmann and Rother, 2018) proposes to simplify the trainable components by making the scene coordinate as the only trainable part of the algorithm. We further simplify their method by replacing the differentiable pose estimation algorithm with a classical one (Kneip et al., 2014) and simply relying on our network ability to accurately predict 3D coordinate predictions. We also reformulate scene coordinate regression as a task of joint globally unique instance segmentation and prediction of local object coordinates (see Section 4.3), which allows us to obtain accurate pose estimates on orders of magnitude larger maps than in (Brachmann et al., 2017; Brachmann and Rother, 2018; Li et al., 2018b; Shotton et al., 2013) despite using training data consisting only of videos traversing environments of interest following a simple trajectory once.

Semantic localization. Semantic information is often incorporated into localization frameworks in one of two ways. Approaches of the first type perform keypoint filtering (Naseer et al., 2017) or feature re-weighting (Kim et al., 2017; Schönberger et al., 2018) of dynamic or difficult objects. Approaches of the second type attempt an explicit fitting of 3D models of individual rooms (Satkin et al., 2012) or buildings (Cohen et al., 2016) or of detailed maps (Chen et al., 2011; Pylvänäinen et al., 2010). The former methods often increase the accuracy of underlying localization algorithms but do not directly address their robustness under changes in the environment. The latter methods are often slow at test time and are more suitable for data collection. A recent work of (Budvytis et al., 2018) attempts to predict a rich representation of per-pixel globally unique instance labels and to show that it is enough to perform localization from it under severe changes in the environment. Our work augments this representation with local coordinate prediction which allows us to obtain 6-DoF pose estimates as opposed to performing image retrieval and to introduce robustness to unseen translation of the camera poses at test time as well as to avoid a computationally expensive step of explicit rotational alignment of label images.

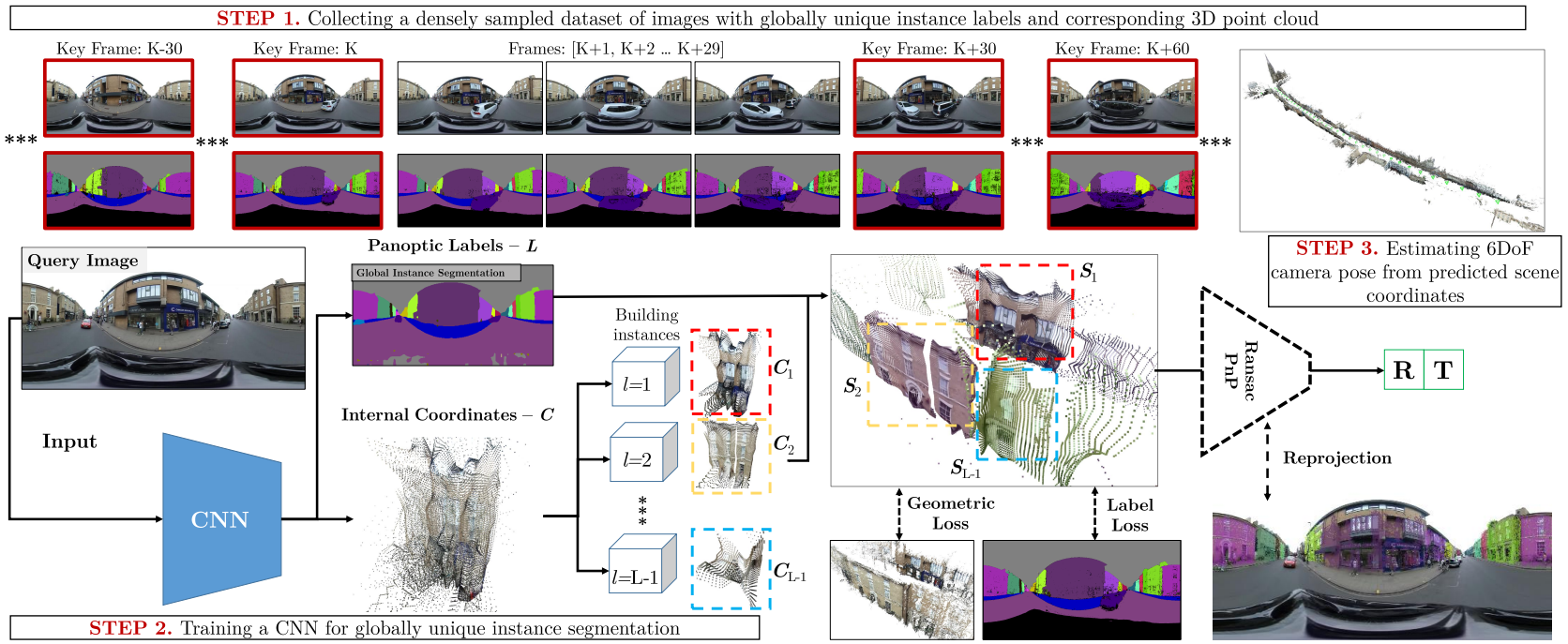


Fig. 4.2 This figure illustrates the three key steps of our method. First, a densely sampled dataset of images is collected and annotated with both class and globally unique instance labels. From these images a 3D point cloud is built using a state-of-the-art library for structure from motion – OpenSFM (map, 2019). Second, a CNN is trained to predict panoptic labels L as well as local PCA whitened coordinates C for each object instance. During the final step predicted local coordinates are unwhitened to obtain corresponding scene coordinates S. EPnP (Lepetit et al., 2009) with RANSAC is used for camera pose estimation.

4.3 Method

Our proposed localization framework consists of three key steps: data collection, training of a CNN to predict globally unique instance coordinates and pose estimation. We illustrate an overview of our method in Figure 4.2. Each step is detailed in one of the sections below.

4.3.1 Data Collection

First, a densely sampled collection of panoramic images of the environment is obtained. Second, a subset of images (e.g. every 30 frames) are hand labelled with both class labels (e.g. sky, road, pedestrian) and globally unique instance labels of buildings[1] and a label propagation algorithm (Budvytis et al., 2017) is used to label the rest of the images. Finally, camera pose estimates and corresponding semantic 3D point cloud are obtained using OpenSFM (map, 2019), an open source structure from motion (SfM) library. Default parameter settings are used unless stated otherwise. The point cloud is projected for each training image using ground truth camera pose in order to produce a 3-channel image containing (x, y, z) coordinates of projected 3D points. When multiple 3D points are projected to the same pixel, the closest one with the same instance label as the source pixel is chosen.

4.3.2 Training

We train a network to jointly predict panoptic segmentation labels and 3D points. The panoptic segmentation labels consists of 10 class labels defined in (Budvytis et al., 2018) which includes road, sky and people as well as instance labels of buildings. For each pixel corresponding to a building we also predicts its 3D coordinates. Towards the goal of improving convergence behavior we normalize ('whitening') the coordinates using principal component analysis (PCA). For a building l consider X_l the set of all 3D points corresponding to that building. We perform PCA on those points and obtain the vector M_l the mean of all points in X_l and W_l the principal component matrix. For a point $S_p \in X_l$ we compute its normalized representation C_p as follows:

$$C_p = (S_p - M_l)W_l \tag{4.1}$$

The network is then trained to predict the whitened representation C_p. We utilize the following identity:

[1]Note that instances of other static objects such as trees or road signs could also be used as demonstrated in (Budvytis et al., 2018).

$$S_p = W_l C_p + M_l \tag{4.2}$$

to obtain real-world coordinates from the whitened representation. We refer to this process as "unwhitening". During inference we obtain the building ID l as a prediction from the segmentation output as visualized in Figure 4.2. Note that this means that a wrongly predicted building ID leads to a very large error of S_p since the wrong mean M_l is added. However for our approach that is not a mayor concern since the RANSAC Algorithm will be able to filter out those outliers. Whitening on the other hand makes significantly easier for the network to fit the target coordinates as shown in Section 4.4. We apply standard cross-entropy loss for both class and instance labels and a Euclidean distance loss for fitting whitened local instance coordinates C_p. Note that for each pixel a $3+L$ dimensional vector is predicted where 3 corresponds to a 3-dimensional coordinate and L corresponds to a total number of panoptic labels.

Training details. Our network consists of a ResNet-50 (He et al., 2016b) backbone followed by bilinear up-sampling with skip layers analogously to FCN (Long et al., 2015). ResNet-50 implementation and initialization weights provided by the PyTorch (Paszke et al., 2017) repository are used. Strided convolution layers are replaced with dilated convolution in order to reduce the down-sampling inside the network. The models are trained for 3000 epochs, unless stated otherwise, using a batch size of 14 (for images of resolution 512×256) and the Adam optimizer (Kingma and Ba, 2014b). The initial learning rate is set to 2×10^{-4} and polynomial decrease (Chen et al., 2018; Liu et al., 2015b) is applied by multiplying the initial learning rate with $((1 - \frac{step}{max\_steps})^{0.9})^2$ at each update step. An L_2 weight decay with factor 5×10^{-4} is applied to all kernel weights and 2D-Dropout (Tompson et al., 2015) with rate 0.5 is used on top of the final convolutional layer. We train all networks on four GeForce GTX 1080 Ti GPUs.

4.3.3 Pose Estimation

Camera pose is estimated with EPnP (Lepetit et al., 2009) perspective-n-point solution with a Random sample consensus (RANSAC) (Fischler and Bolles, 1981b) loop from predicted scene coordinates. Other standard solutions (Kneip et al., 2014) to PnP can be used as well. RANSAC is run for 1000 iterations with points within 0.22° considered as an outlier threshold. Since we aim to recover as accurate 3D coordinates as possible, we do not consider employing differentiable pose estimation algorithms used in (Brachmann et al., 2017; Brachmann and Rother, 2018). The re-projection loss component employed in such algorithms reduces the

accuracy of predicted 3D geometry in favour of a more accurate camera pose estimation. This is demonstrated in Figure 4.7, where scene coordinate prediction accuracy is lower for the loss *L3-Rec-Repr* (40% of points lie within 0.5m of ground truth) which combines reconstruction and re-projection losses than *L2-Rec* (44% of points lie within 0.5m of ground truth) which directly aims at minimizing Euclidean distance between predicted and ground truth scene coordinates.

4.4 Experimental Evaluation

In this section we first discuss the experimental setup and then detail our experimental results.

4.4.1 Experimental Setup

CamVid-360 dataset. CamVid-360 (Budvytis et al., 2018) is a dataset of panoramic videos captured by cycling along the original path of CamVid (Brostow et al., 2009). CamVid-360 training set consists of 7835 images sampled at 30 fps, at resolution 1920×960, which cover two sequences (016E5 and 001TP) of the original CamVid (Brostow et al., 2009) dataset. The query set contains both a test sequence[2] used in (Budvytis et al., 2018) (318 images sampled at 1 fps) as well as a new additional test sequence obtained by downloading Google StreetView panoramic images along the tracks of the original dataset. We estimate ground truth poses for testing images by minimizing the re-projection errors of SIFT (Lowe, 2004) feature matches from the 80 closest images in the training dataset via robust EPnP (Lepetit et al., 2009).

SceneCity dataset. SceneCity (Budvytis et al., 2018) dataset contains images rendered from two artificial cities. See Figures 4.3 to 4.5 and 4.9 for example images and maps. The first city, referred as Small SceneCity, is borrowed from (Zhang et al., 2016). It contains 102 buildings and 156 road segments. The second city, referred as Large SceneCity, contains 827 buildings and 966 road segments in total. The training database consists of 1146 and 6774 images sampled uniformly from each city respectively. 3D point clouds are obtained from Blender directly. Our algorithms are evaluated on two variants of Small SceneCity. For the first variant, 300 camera poses are sampled uniformly from the original track of (Zhang et al., 2016). For the second variant, the same camera poses are used but a random 20% of buildings are removed from the Small SceneCity map. The query set for the Large SceneCity

[2]Note that unlike (Budvytis et al., 2018) we do not use images from sequence 006R0 because this sequence is not covered in training data.

Fig. 4.3 This figure illustration of scene coordinates predicted by CNNs on Camvid with one of five different losses (see Section 4.4 for more details). For visualization the 3D points are coloured with the colour value of the corresponding pixel in image space. The 3D plot is then rotated into birds-eye view to visualize the depth information of our data. L1-Repr visualizes the image when trained with the loss as proposed in (Brachmann and Rother, 2018). The loss lacking one degree of freedom. The result is that the network puts all pixels on a sphere.
The next three models L2 - L4 all predict the 3D world coordinates directly (without "whitening"). The final loss L5 is our model which jointly predicts segmentation and 3D coordinate and uses those for information for instance based whitening. L2 uses a simple quadratic regression loss. It can be seen, that details are distorted compared to the L5 loss. This qualitatively confirms the high accuracy of our approach at small scale. L3 is a linear combination of L1 and L2. The goal of this loss is that the spherical component can help with the detail and remove distortion while the L2 component removes the degree of freedom from the model. L4 is task to jointly predict the segmentation as well as the 3D points, unlike L5 the segmentation is not used for whitening. Our experiments indicate that it is indeed the whitening which is responsible for the performance increase and not the added signal due to the segmentation task.

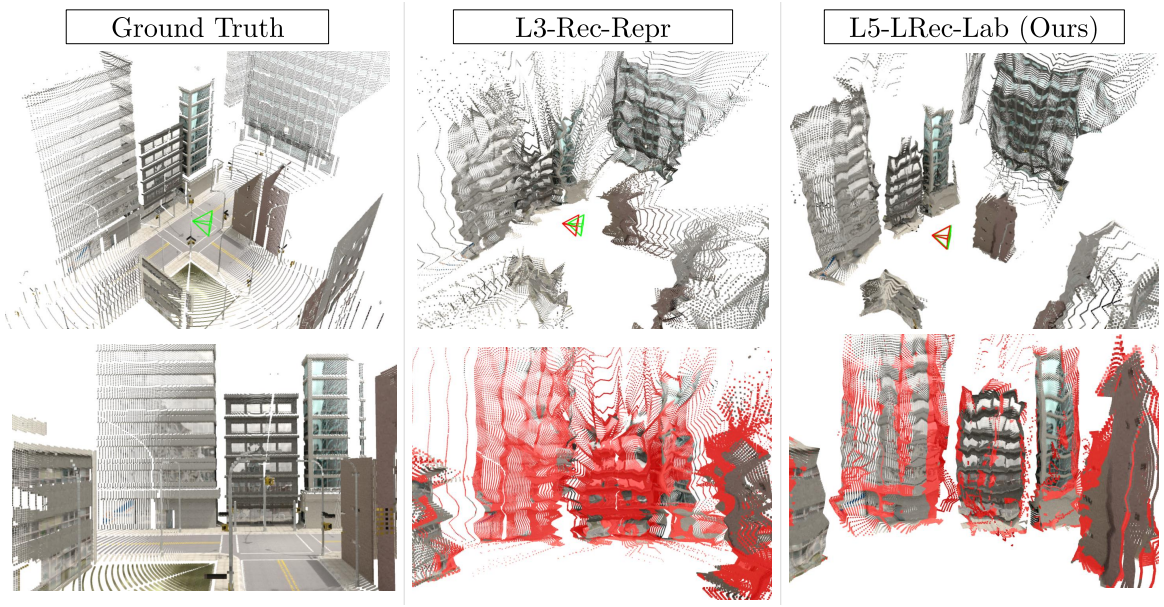


Fig. 4.4 This figure illustrate scene coordinate predictions on the small scenecity datasets. The small scenecity dataset is synthetic dataset more them 8 times the size compared to Camvid dataset. We can clearly observe how the model trained with L3 loss struggles to predict the correct 3D points. Our model on the other hand is able to reconstruct the scene well. The green and red tetrahedron visualize ground truth and predicted pose respectively. We observe that the pose estimation of our model is much more accurate.

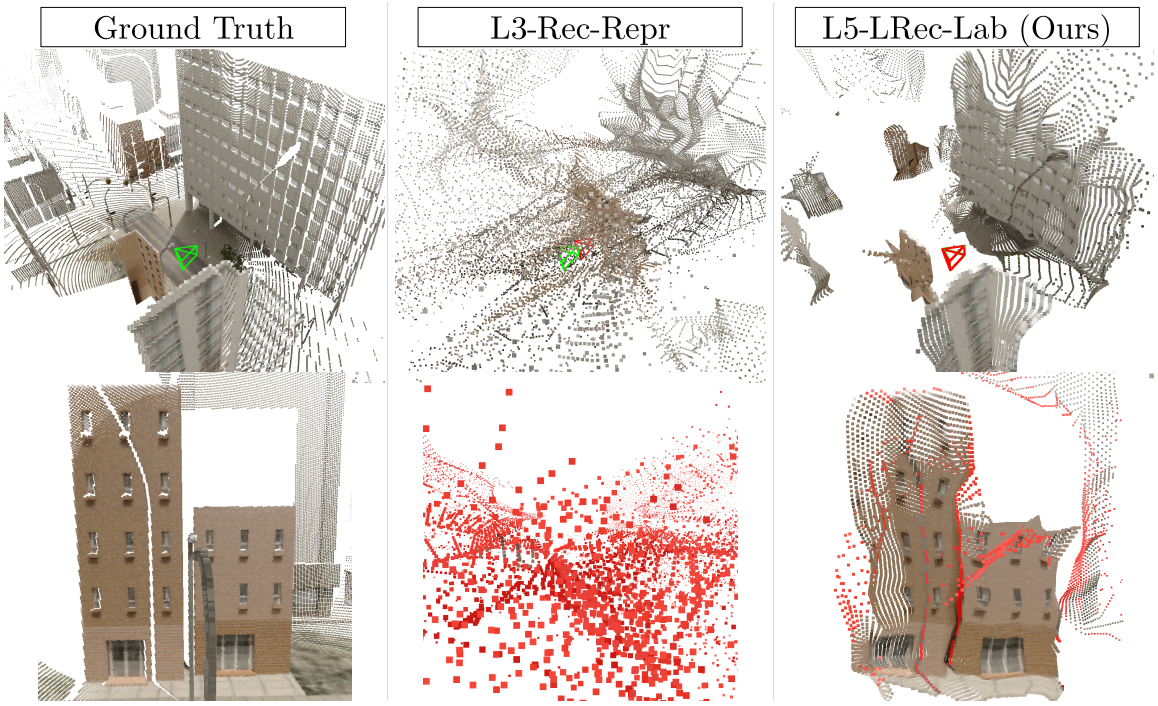


Fig. 4.5 This figure illustrate scene coordinate predictions on the large scenecity datasets. The large scenecity dataset is about 9 times larger then the small scenecity dataset. We observe that model trained with L3 loss is unable to perform 3D coordinate regression. We observe that the model is still able to do a reasonable pose estimation (visualized as green and red tetrahedron). This is due to the fact that the RANSAC algorithm is able to deal with noise and outliers very well. Few reasonable well estimated 3D coordinates are enough in order to obtain a good prediction. Our model on the other hand is able to correctly predict the 3D coordinates of its surroundings, even on large maps. The corresponding pose estimation is very accurate.

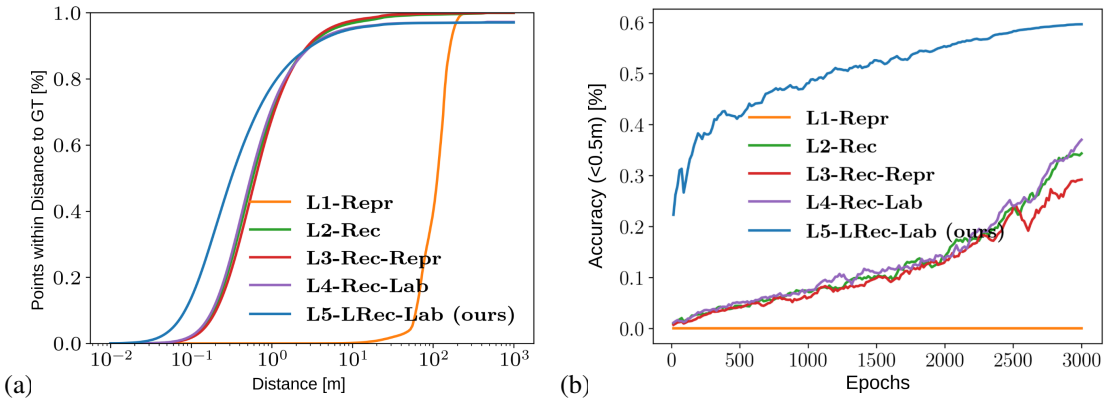


Fig. 4.6 Graph (a) plots the percentage of points for which predicted scene coordinates reside within a given distance of a corresponding ground truth location. Our method (*L5-LRec-Lab*) outperforms alternative approaches at predicting significantly more points with small Euclidean distance. For distances larger than 4m, our method is less accurate. This is due to the error introduced by wrongly predicted building ids. Graph (b) plots the evolution of the percentage of points that reside within 0.5m of the ground truth location as the number of training epochs increases. This is a particularly relevant score since our goal is to localize with half meter accurately. RANSAC will be able to filter out outliers. For a robust solution it is however still crucial to have a substantial amount of prediction within the desired threshold.

consists of 1000 samples near the centre of road segments as explained in (Budvytis et al., 2018).

Evaluation protocol We provide quantitative and qualitative evaluation of the accuracy of both predicted scene coordinates and estimated 6-DoF camera poses. Scene coordinate regression is evaluated by measuring the percentage of points falling within 0.5m, 1.0m and 3.0m of corresponding ground truth targets. We also provide the average distance from ground truth coordinates for all points residing within 3m of their targets. Points further away than 3m are excluded because they correspond to outliers that can obscure the true accuracy of the algorithms evaluated. The accuracy of 6-DoF camera pose estimation is evaluated by measuring median and 95th percentile distance and angular errors between predicted and target cameras. See Figures 4.7 and 4.1 for examples of results.

4.4.2 Experimental Results

Three types of experiments are performed in order to evaluate our proposed framework for joint re-localization and scene understanding. In the first two sets of experiments we evaluate the quality of the scene coordinate prediction and localization respectively. In the final set of

Loss	Scene Coordinate Prediction								GT Mask	Localization							
	CamVid-360 16E5-P2				CamVid-360-StreetView 16E5-P2					CamVid-360 16E5-P2				CamVid-360-StreetView 16E5-P2			
	<3m [%]	M [m]	<1m [%]	<0.5m [%]	<3m [%]	M [m]	<1m [%]	<0.5m [%]		D-Med [m]	A-Med [deg]	D-95 [m]	A-95 [deg]	D-Med [m]	A-Med [deg]	D-95 [m]	A-95 [deg]
L1-Repr	0.00	-	0.00	0.00	0.00	-	0.00	0.00	No	4.95	**1.47**	12.78	5.72	16.88	9.13	135.1	170.2
									Yes	5.49	1.57	14.63	5.14	17.99	7.98	162.43	165.32
L2-Rec	0.9	0.72	0.7	**0.44**	0.58	1.14	0.31	0.12	No	3.6	34.99	210.5	153.9	6.92	40.74	198.2	136.9
									Yes	0.35	1.21	2.59	3.48	0.98	3.62	11.2	54.27
L3-Rec-Repr	0.91	0.76	0.68	0.4	0.56	1.15	0.3	0.1	No	0.63	2.1	4.05	5.23	1.99	7.19	19.5	48.78
									Yes	0.36	**0.95**	2.15	**2.69**	1.15	4.58	37.89	96.03
L4-Rec-Lab	0.9	0.69	0.71	**0.46**	0.66	1.09	0.36	0.14	N\A	0.3	0.96	2.54	4.35	0.9	3.6	12.83	21.61
L5-LRec-Lab (Ours)	0.9	0.48	0.78	**0.63**	0.73	0.75	0.55	**0.33**	N\A	**0.1**	**0.55**	0.64	1.37	0.32	1.62	1.62	7.24

Fig. 4.7 This figure provides a quantitative evaluation of scene coordinate prediction and localization performance for five different losses on the CamVid-360 sequence 16E5-P2 and its StreetView counterpart. For the task of scene coordinate prediction percentages of pixels within 3m, 1m and 0.5m are reported together with average distance for pixels which are within 3m (column M). For the task of localization, median and 95th percentile angular error (A) and camera location distance from ground truth value (D) are reported. For methods which do not explicitly predict instance labels (*L1-L3*), a result which is obtained by masking out pixels which do not belong to building instances (see column GT Mask) is reported for a fair evaluation.

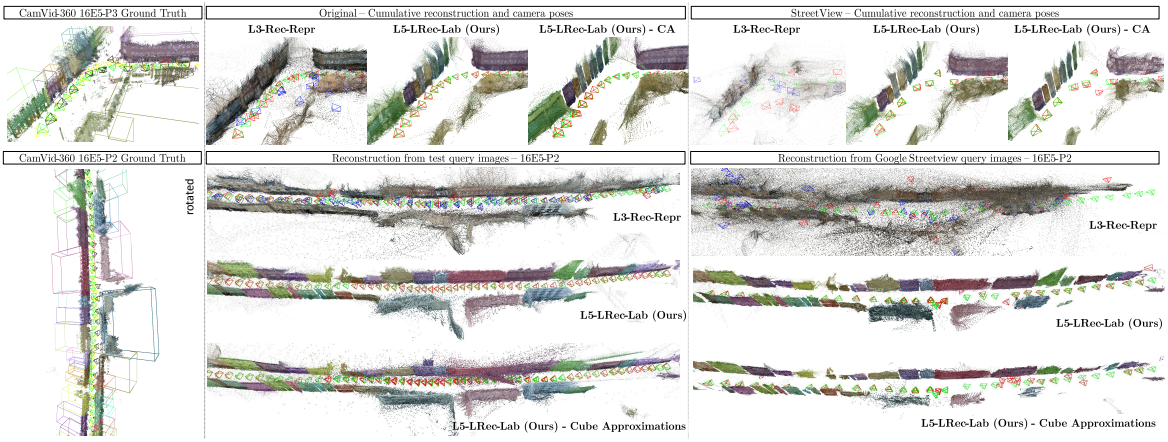


Fig. 4.8 The top left image of this figure displays the ground truth semantic point cloud as well as database (yellow) and query trajectories (green) for the original CamVid-360 16E5-P3 sequence and our collected sequence from Google StreetView images (black). Two groups of three columns at the top show cumulative predicted 3D point clouds (random sample of 1% of total points) with accompanying ground truth camera poses (green) and predicted camera poses (red) for three different methods. Camera poses predicted by PoseNet (Kendall and Cipolla, 2017) are marked in blue. Similar results are provided for sequence 16E5-P2 on the bottom part of the figure. Zoom in for a better view. Also see supplementary material.

Dataset	Scene Coordinate Prediction															Localization													
	L3-Rec-Repr				L5-LRec-Lab (Ours)					L5-LRec-Lab (Ours) – CA						PoseNet				L3-Rec-Repr (M)				L5-LRec-Lab (Ours)				L5-LRec-Lab (Ours) – CA			
	<3m [%]	M [m]	<1m [%]	<0.5 [%]	<3m [%]	M [m]	<1m [%]	<0.5 [%]	IoU [%]	<3m [%]	M [m]	<1m [%]	<0.5 [%]	IoU [%]		D-Med [m]	A-Med [deg]	D-95 [m]	A-95 [deg]	D-Med [m]	A-Med [deg]	D-95 [m]	A-95 [deg]	D-Med [m]	A-Med [deg]	D-95 [m]	A-95 [deg]	D-Med [m]	A-Med [deg]	D-95 [m]	A-95 [deg]
CamVid-360	0.82	1.1	0.44	0.18	0.88	0.6	0.72	**0.53**	**0.5**	0.78	0.95	0.49	0.25	0.46		12.4	15.6	106	118	0.73	1.66	4.86	11.9	0.22	0.71	**1.14**	**3.55**	0.39	1.16	2.64	6.6
CamVid-360 StreetView	0.35	1.43	0.12	0.03	0.62	0.94	0.4	0.2	0.28	0.38	1.27	0.17	0.05	0.19		45.2	79.8	163 (91.5)	172 (150)	3.99	9.29	96.2 (22.87)	156.86 (39.43)	**0.75**	**2.32**	82.5 (3.01)	113 (6.93)	2.12	5.28	148 (32.9)	160 (53.2)
SceneCity - Small	0.79	1.57	0.18	0.03	0.8	1.22	0.37	0.11	0.56	0.75	1.14	0.37	0.13	0.59		1	1.94	3.78	8.99	0.63	0.95	1.25	**2.02**	0.48	1.12	1.17	**3.18**	0.32	0.86	**0.82**	**1.82**
SceneCity - Small Missing Buildings	0.54	1.64	0.11	0.02	0.64	1.23	0.29	0.08	0.39	0.6	1.17	0.3	0.1	0.41		5.63	4.37	93.2	91.8	0.98	1.92	2.89	5.5	0.64	1.59	1.63	4.54	0.46	1.13	1.41	3.2
SceneCity Large	0.08	2.03	0.01	0.00	0.83	0.78	0.61	**0.39**	**0.34**	0.71	1.07	0.4	0.18	0.32		2.69	5.95	52.7	44.9	9.07	4.27	58.1	30.2	0.2	0.76	**0.59**	**1.86**	0.27	1.01	0.84	2.84

Table 4.1 This table provides a quantitative evaluation of our approach on large datasets of CamVid-360 and SceneCity. Similar metrics are used as in Figure 4.7. A method based on joint reconstruction and re-projection losses *L3-Rec-Repr* is not able to accurately fit large maps. Our method demonstrates superior performance with more than 39% of points predicted within 0.5m on Large SceneCity. Relatively poorer reconstruction quality in Small SceneCity dataset (compared to Large SceneCity) can be explained by a higher density of tall buildings, as the accuracy of the 3D points drops significantly for the tops of buildings. Our method *L5-Rec-Repr (Ours)* outperforms both PoseNet (Kendall and Cipolla, 2017) and *L3-Rec-Repr* in all experiments, with an exception of the angular distance error on the Small SceneCity dataset due to the effect of re-projection loss component in *L3*. However, a version of our method which uses approximate 3D maps outperforms *L3*. This can be explained by cuboids being a good approximation of artificial buildings. Also note that numbers in brackets correspond to 80th percentile distance and angular errors for CamVid-360 StreetView dataset.

experiments we explore the feasibility of performing localization by using highly compact and fast-to-query maps that are made of cuboids approximating buildings.

Scene coordinate regression. Firstly, we compare five CNNs trained with different losses and evaluate their performance on the task of scene coordinate regression on a subsequence 16E5-P2 of CamVid-360 (Budvytis et al., 2018) dataset. The first loss *L1-Repr* minimizes a re-projection error of points on a spherical image plane: $L1\text{-}Repr(S,S^{gt}) = \sum_{p\in P} || \frac{R^T S_p + T}{||R^T S_p + T||} - V_p ||$. Here R and T are ground truth camera rotation and translation matrices, P – a set of all pixels in an image, S_p – predicted scene coordinates for a pixel p and V_p is a vector pointing to pixel p projection on a spherical image. It is a straightforward adaptation of a re-projection loss used in (Brachmann and Rother, 2018) from planar images to spherical images. It is also equivalent to a loss proposed in (Li et al., 2018b). The second loss $L2\text{-}Rec = \sum_{p\in M^{gt}} ||S_p - S_p^{gt}||$ directly minimizes the Euclidean distance between predicted scene coordinates S_p and ground truth scene coordinates S_p^{gt} for all pixels for which ground truth coordinates are available – set M^{gt}. Depending on choices of learning rates and relative loss weighting parameters the first two stages of the approach of (Brachmann and Rother, 2018) can be viewed as a mixture of both aforementioned losses. We approximate this work by loss $L3\text{-}Rec\text{-}Repr = \alpha L1\text{-}Rep + (1-\alpha) L2\text{-}Rec$, where α is set empirically to 0.02. Note that we do not use the third stage of differentiable pose prediction, and use a classical

method (Lepetit et al., 2009) instead. Also note that the authors of (Brachmann and Rother, 2018) report only a small advantage of using this stage at a cost of the lack of convergence on Street scene of Cambridge Landmarks (Kendall et al., 2015) dataset. The final two losses considered are *L4-Rec-Lab* and *L5-LRec-Lab*. The former combines a standard cross-entropy loss used for semantic segmentation and a reconstruction loss *L2*, whereas the latter combines a cross-entropy loss with reconstruction loss in local whitened coordinate space. We empirically set relative weighting between cross-entropy loss and reconstruction losses to 0.1 and 0.5 respectively. As shown in Figures 4.3, 4.6(a) and 4.7, re-projection loss (*L1-Repr*) alone does not enable a CNN to recover accurate geometry and instead predict a point cloud of an approximately spherical shape. In contrast, using methods that directly predict scene coordinates (*L2*, *L3*, *L4*) lead to high accuracies with more than 40% of points residing within 0.5m of their ground truth location. Our method *L5-LRec-Lab* outperforms the alternatives by more than 17%. This is due to a faster convergence at train time, which is caused by a simpler optimization task resulting from the separation of object centre and local coordinate prediction (see Figure 4.6(b)). The difference in performance between the aforementioned methods becomes even bigger when larger maps such as full CamVid-360 or artificial cities are considered, as shown in Figures 4.4, 4.5 and 4.8 as well as Table 4.1. We conclude that especially in large maps our join training and normalization approach is able to add a significant benefit to the model performance.

The quality of approximate depth recovered is reduced by the fact that training images are collected from a single cycling track that has little variation in two dimensions. This explains the difference between our results and the relatively good performance in recovering approximate depth reported in (Brachmann and Rother, 2018). In contrast, loss based on 3D coordinate regression produces high quality reconstruction, but due to the lack of semantic information it presents a lot of outliers. This problem is removed with loss L4-3D-Lab.

The loss L5-LRec uses instance based whitening as discussed in Section 4.3.2. Our experiments indicate that using whitening leads to much better convergence behavior (Figure 4.6 (b)) and better performance on the sub-meter scale (Figure 4.6 (a)). More then 50% of examples are within half a meter from its true position on the Camvid dataset. However computing the unwhitened world coordinate S_l relies on correctly predicting of the building id l for the corresponding pixel. If a wrong *id* is predicted S_L will be placed on a different building which can be several hundred meters away. This effect can be observed in Figure 4.6 (a), where we observe that the performance of our model plateaus at the meter scale. This is however not an issue for our overall goal of performing localization. The RANSAC algorithm used to solve the PnP task is robust to outliers. Outliers are filtered out during the RANSAC iteration and will not influence the final result.

Localization. As with scene coordinate regression, we first evaluate localization accuracy of various methods on the sequence 16E5-P2 of CamVid-360 (Budvytis et al., 2018) dataset in detail. We then follow by experiments on full length sequences of CamVid-360 and two artificial cities. It can be seen in Figures 4.6(c) and (d) that CNN trained using *L1-Repr* loss shows a poor performance in estimating 3D location, but a relatively low angular error. *L2-Rec* and *L3-Rec-Repr* perform poorly at both tasks if pixels not belonging to building instances are not masked out at test time. Note that we use ground truth masks in order to evaluate the upper bound of the performance of both methods. Directly predicting semantic scene coordinates (loss *L4-Rec-Lab*) produces performance similar to masked versions of *L2* and *L3* because its localization accuracy is limited by the accuracy of 3D coordinates predicted. Hence it is not surprising that our proposed method based on predicting local object coordinates (*L5-LRec-Lab*) significantly outperforms all the alternative methods at both angular error and camera location distance error. Similar trends are observed on large experiments, as reported quantitatively in Table 4.1 and qualitatively in Figures 4.8 and 4.9. Also note that while PoseNet (Kendall and Cipolla, 2017) (a standard setup[3] with geometric re-projection error and ResNet-50 encoder), adapted to performing on panoramic images, shows a seemingly competitive performance on Small and Large SceneCity data, its performance drops on the CamVid-360 dataset. This can be explained by PoseNet (Kendall and Cipolla, 2017) sensitivity to over-fitting to the training images, because they are obtained from a single video as opposed to a diverse set of images. This is also supported by a significant drop in accuracy on Small SceneCity images with missing buildings. Our method significantly outperforms both PoseNet and L3-3D-Repr (Brachmann and Rother, 2018) in mean distance and angular errors when all predictions are considered. When predictions further away than 3m are excluded, the difference between the approaches is reduced. Nevertheless, we significantly outperform competing approaches owing to a much higher quality of predicted 3D maps. Note that relatively high mean distance and angular errors are mainly appearing in sequence 016E5-P1, where the buildings are relatively small and far away. Also note that our approach is significantly less sensitive to changes in the environment such as missing buildings in Small SceneCity map (see Figure 4.1).

Localization in approximate maps. In the final set of experiments we explore the alternative of predicting a 6-DoF pose from simplified approximate 3D maps. We compare reconstruction and localization accuracies using hand-extracted 3D cuboids from points clouds. As expected, on CamVid-360 dataset, scene coordinate and camera pose prediction

[3]Note that in order to use PoseNet (Kendall and Cipolla, 2017) on equirectangular images an explicit rotation augmentation of the camera pose needs to be performed for each crop. We limited crops to a horizontal shift only which corresponds to the rotation of the camera around its axis.

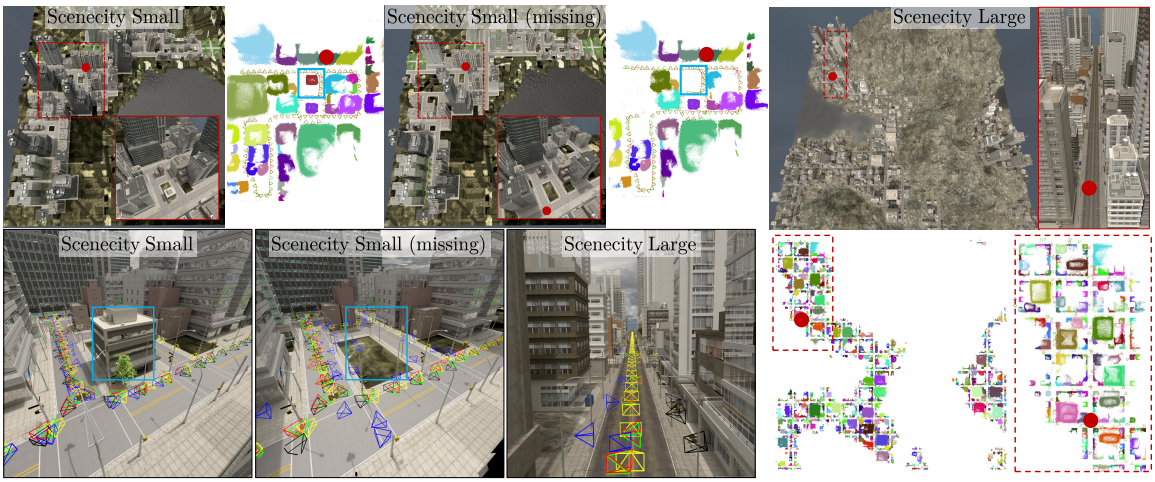


Fig. 4.9 The top row of this figure shows artificial city maps using a top-view orthographic projection. Each city view has a region zoomed in and visualized from a different angle and a corresponding 3D point cloud obtained by accumulating 3D points predicted from test images. Examples of missing buildings are marked in blue rectangles. Three images at the bottom left illustrate the view seen by a camera, whose position is marked as a red dot. They show camera poses of ground truth database (yellow), ground truth query (green), *L3-Rec-Repr* (black), PoseNet (Kendall and Cipolla, 2017) (blue) and *L5-LRec-Lab (Ours)* (red). Zoom in for a better view. Also see supplementary material.

is of lower accuracy than that of the network trained using a precise 3D map, as shown in Figures 4.8 and 4.1. However, on the Small SceneCity data, this network outperforms *L5-LRec-Lab (Ours)*. This can be explained by cuboids being a good approximation for buildings in artificial cities. Moreover, higher performance in localization is obtained than competing approaches of PoseNet (Kendall and Cipolla, 2017) and *L3-3D-Repr* in all experiments. This is a highly encouraging result which, in the future, may alleviate the need of a computationally expensive step of building 3D point clouds of cities.

4.5 Conclusions

In this chapter we saw how we can utilize geometric structure to improve performance. We did this by solving 3D scene coordinate regression as an auxiliary task first, which then allows us to use geometric modelling to solve the pose regression task. Our approach is robust to changes in the scene, as shown with the missing building experiment. Unlike direct coordinate regression (like Posenet Kendall et al. (2015)) our approach is able to generalize to new positions in the scene as long as it is part of the known world map. Our approach therefore solves the issue discussed in Section 2.1. In addition, the semantic output of our model is much more interpretable than simple position coordinate regression. Our model predicts the entire scene in multiple modalities, which allows close inspection of which part

of the model works well and what are the potential failure cases. Lastly our method is a good example of how multiple tasks can assist each other rather then just co-exist when utilizing the underlying structure of the problem.

Limitations & further work: Our method works best on small to medium-size maps. For autonomous driving applications it would be beneficial to be able to localise on city scale maps. A very important step for further research would therefore be to modify the algorithm to work well on larger maps. At the moment, the main bottleneck is the cross-entropy loss, which requires the network to learn millions of building IDs in a full-scale city. Designing an approach that is able to handle this will require us to solve some practical issues such as an infeasibly large memory footprint when implementing the cross-entropy naively. Possible solutions would include embedding the building IDs into a geometrical space and learning them using a hinged regression loss. An alternative solution would be to encode the building IDs into a binary vector and learning to predict the vector as a multi-class classification problem. A different approach would be to move away from building IDs and instead perform classification at block or neighbourhood level. An additional advantage of this idea would be that it also gets rid of the need to manually label every building.

Chapter 5

Beating State of the Art Performance with Structured Modelling

In this chapter we show that the structured modelling approach is able to improve on a well studied task on an existing established benchmark. This shows that the method can be relevant in practical machine vision tasks, being able to compete with, and beat, many other great ideas and approaches.

This chapter is largely based on our Detect-to-Retrieve (Teichmann et al., 2019) paper which was published at the Conference on Computer Vision and Pattern Recognition (CVPR) 2019 conference and is a joint work with Andre Araujo. In this work we tackle the challenging landmark recognition problem. We compare our results on the established Oxford and Paris datasets where we improve upon the state-of-the-art by significant margins.

We achieve this by combining global semantic knowledge obtained from object detection with geometry aware local feature descriptors. We name our approach regional descriptors and we are able to show that it is able to effectively combine the advantages of global and local descriptors.

5.1 Landmark Recognition and Image Retrieval

The goal of landmark recognition is to identify an entity of interest ("Landmark"), given an image. Typical landmarks are structures (e.g. buildings, bridges and churches), monuments or natural landmarks (mountains, lakes and waterfalls). Landmarks can also be a collection of objects; for example the "Skyline of New York" is considered a landmark. The defining feature of a landmark is that it is a named and well recognized instance which many people

(a) Tower Bridge, London (b) King's College Chapel (c) Examples from the Google Landmarks dataset

Fig. 5.1 Examples of Landmarks. Note that landmark recognition is an instance level task. A system which is given Figure 5.1(a) as input is asked to output "Tower Bridge, London" as answer, unlike classification where "Bridge" would be a sufficient answer.

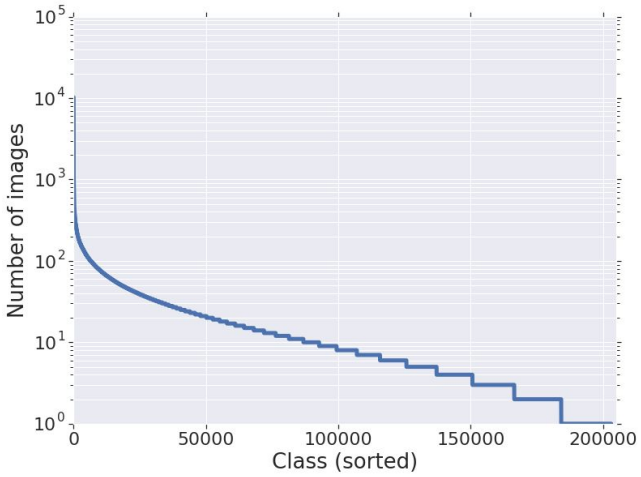
Category	Structures	Monuments	Natural landmarks	Collection of Objects
Subcategories	Buildings	Statues	Mountains	Places
	Bridges	Memorials	Waterfalls	Skyline
	Churches	Artwork	Lakes	
	(City) Walls		Meadows	

Table 5.1 A non-exhaustive list of landmark categories.

find interesting enough to take pictures of. A non-exhaustive list of landmark categories is given in Table 5.1. Some examples of landmarks are depicted in Figure 5.1.

An important aspect of landmark recognition is that it is an instance level task. Consider Figure 5.1, a system which which is given Figure 5.1(a) as input is asked to output "Tower Bridge London" as answer. A system which classifies the input as an image of a Bridge is not enough. Similarly image Figure 5.1(b) should be identified as King's College Chapel and not just as Chapel or Church.

Landmark recognition is related to the well studied image classification task. In particular, landmark recognition can be modelled as a classification problem, where each individual landmark is a class. This however is not practical for typical landmark datasets, due to the large number of classes and the long-tailed nature of the label distribution. Consider the Google Landmark dataset (Bohyung Han, 2019). It consists of more then 5 million images containing over 200 000 unique landmarks. This is much more than typical computer vision classification tasks have, which commonly contain up to 1000 classes. In addition to this, the distribution of labels is very long-tailed Figure 5.2. While there are 25 images per class on average, more than half of the landmarks are shown on less than 10 images. For almost one in ten landmarks there is only one example available. In practice this means that our system also has to implicitly be able to solve the one-shot learning problem.



Number of images	Proportion of classes with less or eq. images
1	9%
5	38%
10	57%
25	80%
100	93%

Fig. 5.2 Number of images for each landmark. It can be seen that the distribution is very long-tailed. Almost one in ten landmark is only shown in one image and half of the landmarks are shown in less then 10 images. On the other hand there are some landmarks with up to 10 000 examples.

This is why landmark recognition is commonly formulated as an image retrieval task. In image retrieval, we have a large database of images. Given a query, the goal is to retrieve images that are semantically similar to the query. Note that there is a dualism between classification and image retrieval. Each task can be solved given a perfect algorithm for the other. A practical difference between the two tasks is that most image retrieval systems try to identify low level discriminators. These are features that are unique to each instance and help to tell semantically similar objects apart. Most classification pipelines, however, utilize high-level semantic clues towards the goal of identifying a broad amount of objects from the same class.

5.2 Methods for Image Retrieval

Image retrieval systems are typically consist of two components, firstly a feature descriptor which processes an image and produces a feature vector. Usually both the database and the query image are processed by the same feature descriptor. The second component compares the features of a given query image with the features of images in the database. It produces a ranked result of potential matches. This result is traditionally re-ranked by more expensive post-processing algorithms (Noh et al., 2017a). A schematic example of a typical image retrieval system is given in Figure 5.3.

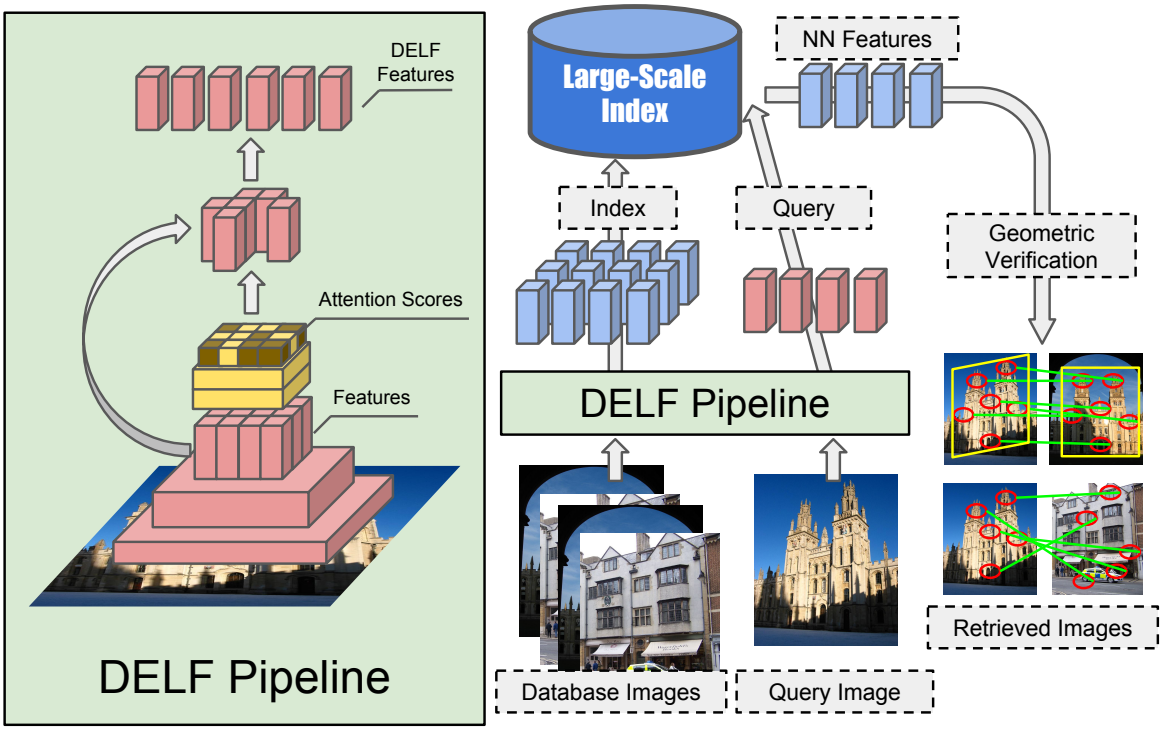


Fig. 5.3 A typical image retrieval pipeline. Local features are computed for all database images offline. During inference, the features are computed for the query image and compared to the features in the database. The results are filtered and ranked using geometric verification. Traditionally, local features have been computed using SIFT like algorithm. Noh et al. (2017a) proposed to use neural networks for this step and named their approach DELF (Deep Local features).

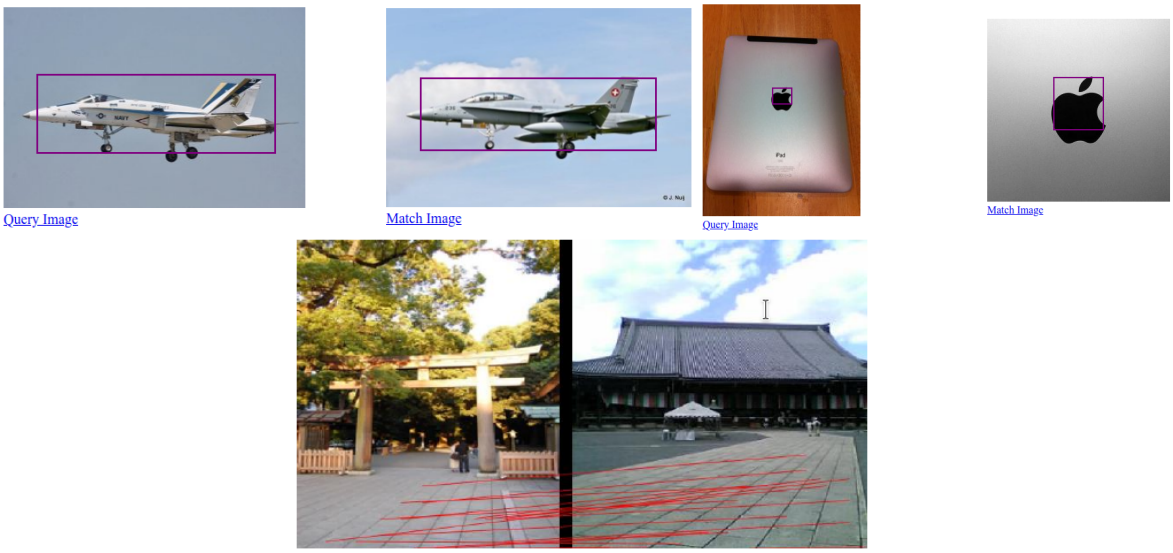


Fig. 5.4 Examples of failure cases of local descriptors. Local descriptors lack semantic understanding, and thus are easily fooled by matching shapes. This can lead to high-scoring false positives.

Local descriptors: Most traditional systems (Bay et al., 2008; Buddemeier and Neven, 2012; Lowe, 2004) rely on hand-crafted local feature descriptors like Scale-invariant feature transform (SIFT). Variants of SIFT such as RootSIFT (Arandjelović and Zisserman, 2012) and Hessian-affine feature (Perd'och et al., 2009) still achieve state-of-the-art performance. DELF (Noh et al., 2017a) (see fig. 5.3) is a system which uses neural networks to produce local features.

Local feature descriptors are particularly suited for large-scale datasets in natural environments. Geometric reasoning can be applied to detected local features. This makes them very robust to changes in camera position and angles. There weakness, however, is a lack of semantic understanding. This can produce high-scoring false positives when local features align. Examples of this are depicted in Figure 5.4. Non-geometric image transformation such as day and night cycles can also impact performance.

Global descriptors: CNNs have been successfully used as global feature descriptors. For this the output of some layer in a network such as ResNet (He et al., 2016c) is used as neural code to semantically describe an entire image. For this, off-the-shelf networks trained in the ImageNet task (Deng et al., 2009b) have successfully been used. In addition a variety of methods have been proposed to further fine-tune the networks for specific datasets (Arandjelović et al., 2016; Gordo et al., 2016; Hadsell et al., 2006; Radenović et al., 2016; Wang et al., 2014). For this, triplet loss (Wang et al., 2014) is commonly used. Pooling

is commonly applied to the spatial dimensions of the output of those networks. This reduces the feature size and adds spatial invariance to the descriptor.

CNN based global descriptors are able to process high-level semantic clues. This makes them resilient to local distractors, overcoming the main weaknesses of local descriptors. However, since they compute one representation for the entire scene, they generalize badly to geometric transformations such as viewpoint changes and object sizes.

Global descriptors have shown some very promising results on smaller datasets depicting landmarks under ideal conditions (Philbin et al., 2007, 2008a; Radenovic et al., 2018). However, they still lag behind in performance when used on large-scale datasets and datasets where the landmark is not the most prominent object in the image (Bohyung Han, 2019; Noh et al., 2017c; Radenovic et al., 2018).

Regional descriptors: Our method, detect-to-retrieve, aims to marry the benefits of local and global descriptors by utilizing semantic-aware region proposals. To the best of our knowledge nobody has tried to design semantic-aware regional descriptors. However, utilizing region selection as part of image retrieval systems has been explored under the framework of regional search and aggregation: (i) regional search: selected regions are encoded independently in the database, allowing for retrieval of subimages; (ii) regional aggregation: selected regions are used to improve image representations. In the following, we review these two types of approaches.

Regional search. Many papers propose to describe regions using VLAD (Jégou et al., 2010) or Fisher vectors (Jégou et al., 2012): Arandjelovic and Zisserman (Arandjelovic and Zisserman, 2013) use a multi-scale grid to extract 14 regions per image; Tao et al. (2014) use selective search Uijlings et al. (2013) with thousands of regions per image; Kim et al. (2015) use maximally stable extremal regions (MSER) Matas et al. (2004). Razavian et al. (2016) use a multi-scale grid with 30 regions per image, and compute the similarity of two images by taking into account the distances between all region pairs. Iscen et al. (2018, 2017) leverage multi-scale grids in conjunction with CNN features (Radenović et al., 2016), to enable query expansion via diffusion. More recently, Radenović et al. (2018) performed a comprehensive evaluation of retrieval techniques and concluded that existing regional search methods may improve recognition accuracy, but with significantly larger memory and complexity costs. In contrast, our detect-to-retrieve framework aims at efficient regional search via the use of a custom-trained detector.

Regional aggregation. Tolias et al. (2015b) leverage the grid structure from Razavian et al. (2016) to pool pre-trained CNN features (Krizhevsky et al., 2012a; Simonyan and Zisserman, 2015) into compact representations; approximately 20 regions are selected per

image. Radenović et al. (2016) build upon (Tolias et al., 2015b) by re-training features on a dataset collected in an unsupervised manner. Gordo et al. (2016) train a region proposal network (Ren et al., 2015a) from semi-automatic bounding box annotations, to replace the grid from (Tolias et al., 2015b). Hundreds of regions per image are considered in this case. Our work departs from these papers by using a small set of regions (fewer than 5 per image), and by formulating regional aggregation as a new match kernel (instead of regional sum pooling as in (Gordo et al., 2016; Tolias et al., 2015b)).

In addition, all of the methods above have in common that they use very simple region selection heuristics. Usually regions are selected uniformly in a grid. Thus the region selection does not add any semantic knowledge to the descriptors. In particular, the issues shown in Figure 5.4 are not solved using those methods. To the best of our knowledge our work is the first to propose semantic aware regional descriptors, which are able to tackle the issues discussed in Figure 5.4.

5.3 Detect-to-Retrieve

In our work (Teichmann et al., 2019) we propose regional descriptors as a way to merit the benefits local and global descriptors. Towards this goal we use an instance agnostic landmark detector, which predicts regions in the image corresponding to possible landmarks. The detector is only challenged with the task of identifying where in the image a landmark is, but not to recognize the landmark instance. Thus the detector can utilize the entire dataset to learn semantically meaningful features which generalize well across many different landmarks. Local descriptors are then computed inside the proposed regions and stored in the index.

The detector has the global view of the entire image and is able to steer the local descriptor towards semantically meaningful regions. Our system is therefore able to avoid local distractors as shown in Figure 5.4. An overview of our method is illustrated in Figure 5.5.

5.3.1 Google Landmark Boxes Dataset

In order to train a meaningful landmark detector we need a large-scale dataset with box labels for landmarks. To the best of our knowledge, no manually curated datasets of landmark bounding boxes exist. Gordo et al. (2016) use SIFT (Lowe, 2004) matching to estimate boxes in landmark images. Such boxes are biased towards the feature extraction and matching technique, and may contain localization errors. Their dataset contains 49k boxed images, from 586 landmarks. This is too small for our use case. The OpenImages dataset (Kuznetsova

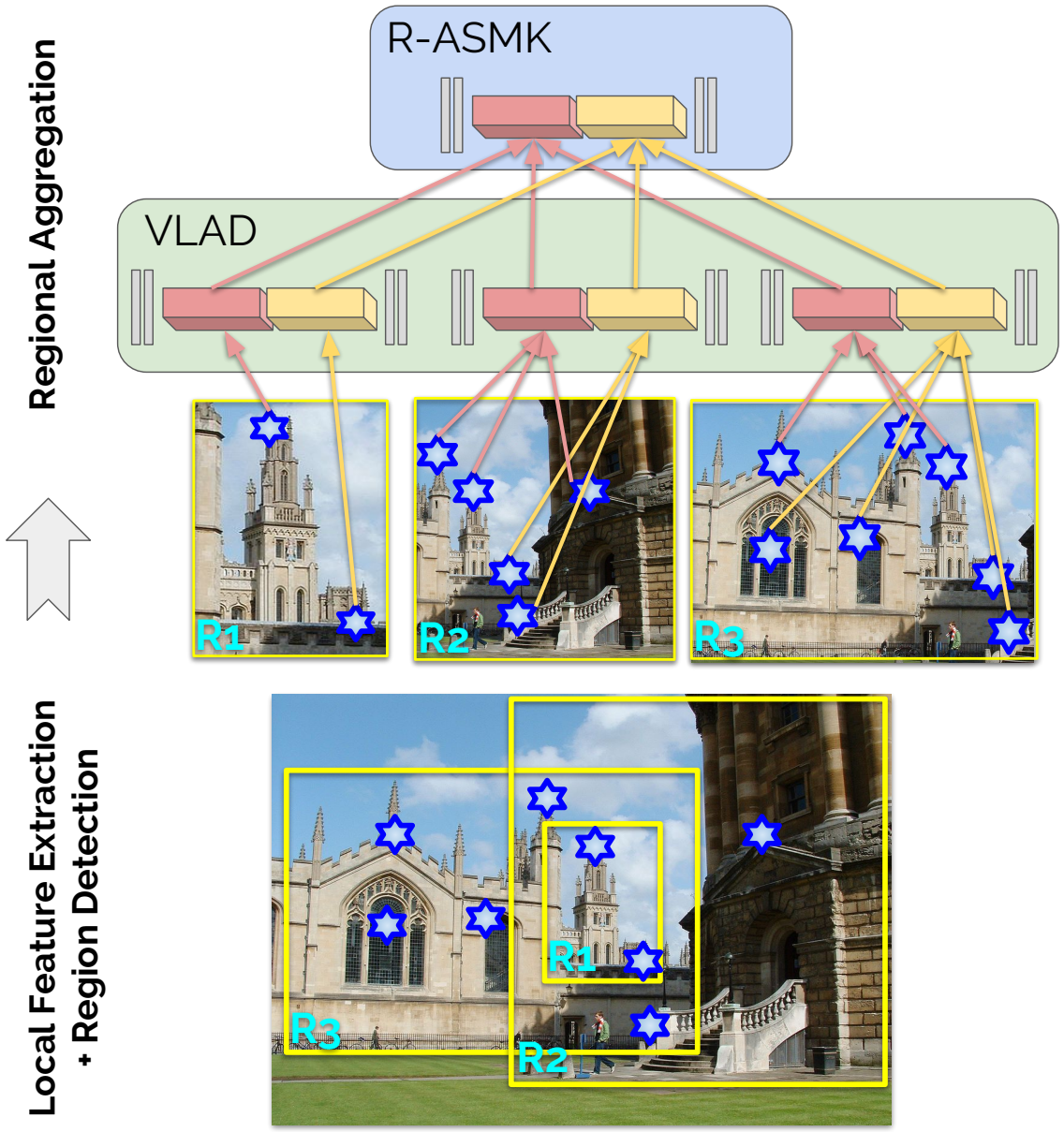


Fig. 5.5 Overview of our proposed regional aggregation method. Deep local features (stars) and object regions (boxes) are extracted from an image. Regional aggregation proceeds in two steps, using a large codebook of visual words (red and yellow visual words are depicted): first, per-region VLAD description; second, sum pooling and per-visual word normalization. Our final regionally aggregated image representation can be combined to selective match kernels and provide improved image similarity estimation: we refer to this technique as regional aggregated selective match kernels (R-ASMK). It leverages detected regions to improve image retrieval with no dimensionality increase when compared to the original ASMK method (Tolias et al., 2015a).

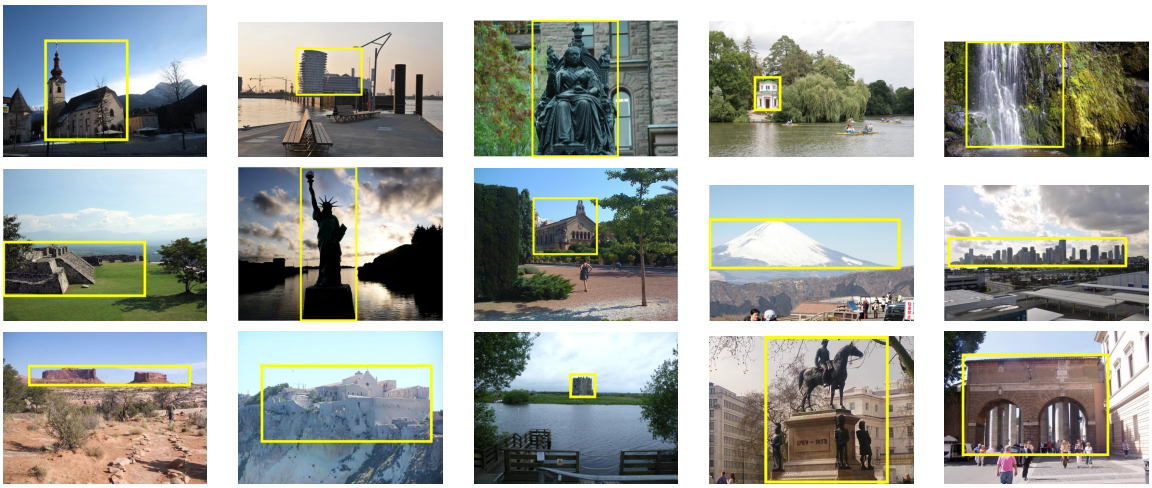


Fig. 5.6 Examples of annotated images from our Google Landmark Boxes dataset. A box is drawn around the most prominent landmark depicted in the image. The dataset contains a wide variety of objects, ranging from man-made to natural landmarks.

et al., 2018) contains 9M images, annotated with generic object bounding boxes. Some of them may be considered landmarks, such as buildings, towers, skyscrapers and billboards. However, these classes represent a small fraction of potential landmarks. Owing to the lack of alternatives we decided to create our own dataset with annotated landmarks. We utilized human raters to annotate the regions of interest, and produced 86k boxed images from 15k landmarks. The boxes are now publicly available as part of the Google Landmarks dataset (Bohyung Han, 2019).

One of the main challenges in such a fine-grained dataset is the inherent long tail in the number of image samples per class. In GLD, some landmarks are associated to several thousands of images, while for about half of the classes, only 10 or fewer images are provided. Our goal is to represent landmarks in a balanced manner in our new dataset, such that trained detectors are able to localize a wide variety of objects. For this reason, we first separate part of the 1.2M training set into a validation set. We randomly select four training and four validation images per landmark. In total, this yields 58k and 36k boxed images for training and validation, respectively. Note that this means that for about 40% of landmarks, all available images are annotated.

Examples of annotated images are shown in Figure 5.6. In some cases, it is not possible to identify a prominent landmark (see Figure 5.7): the landmark of interest may be occluded, or the image may actually show the surroundings of a landmark. We remove such corner cases from our dataset (this applied to about 8% of images which were initially selected), leading to a final dataset with 54k and 32k boxed images for training and validation, respectively.

Fig. 5.7 Examples of Google Landmarks dataset images that do not depict a prominent landmark. In such cases (about 8% of images), no boxes were drawn, and the images were not included in the Google Landmark Boxes dataset.

5.3.2 Regional Search and Aggregation

We present techniques that enhance image retrieval performance by utilizing bounding boxes predicted by a trained landmark detector. In particular, our approach builds on top of deep local features (DELF) (Noh et al., 2017c) and aggregated selective match kernels (ASMK) (Tolias et al., 2015a), which were recently shown to achieve state-of-the-art performance on a large-scale image retrieval benchmark (Radenović et al., 2018).

Background

We briefly review the aggregated match kernel framework by Tolias et al. (2015a). A matching kernel $K(X,Y) \rightarrow \mathbb{R}$ is a function of two images X and Y which is designed to be positively correlated with the similarity between X and Y. That is two images showing the same or a similar object are supposed to have a high K score.

An image X is described by a set of m vectors each of dimension d: $\mathscr{X} = \{x_1, x_2, \ldots, x_m\} \subset \mathbb{R}^d$. The set $\mathscr{X}$ is called "local descriptors". A codebook $\mathscr{C} = \{c_1, c_2, \ldots, c_k\} \subset \mathbb{R}^d$ consists of k vectors which are called "visual words". The codebook is usually obtained by computing k-means on the joined set of local descriptors from the training images. The set of cluster centroids is then used as codebook, i.e. each cluster centroid is one visual word. Consider a nearest neighbor quantizer

$$\begin{aligned} q : \mathbb{R}^d &\rightarrow \mathscr{C} \subset \mathbb{R}^d \\ x &\longmapsto q(x), \end{aligned} \tag{5.1}$$

which assigns each descriptor to the closest visual word. For a given image X and its set of local descriptors $\mathscr{X}$ we denote $\mathscr{X}_c = \{x \in \mathscr{X} : q(x) = c\}$ as the subset of descriptors from X, which are assigned to visual word c.

According to the framework proposed by Tolias et al. (2015a), the similarity between two images X and Y, represented by local descriptor sets $\mathscr{X}$ and $\mathscr{Y}$, can be computed as:

$$K(X,Y) = \gamma(\mathscr{X})\gamma(\mathscr{Y}) \sum_{c\in\mathscr{C}} \sigma(\Phi(\mathscr{X}_c)^T \Phi(\mathscr{Y}_c)) \tag{5.2}$$

where $\Phi(\mathscr{X})$ is an aggregated vector representation, $\sigma(.)$ denotes a scalar selectivity function and $\gamma(\mathscr{X}) = \left(\sum_c \sigma(\Phi(\mathscr{X}_c)^T \Phi(\mathscr{X}_c))\right)^{-1/2}$ is a normalization factor. This formulation encompasses popular local feature aggregation techniques, such as Bag-of-Words (Sivic and Zisserman, 2003), VLAD (Jégou et al., 2010) and ASMK (Tolias et al., 2015a).

In particular, for VLAD we set $\sigma(u) := u$ and

$$\Phi(\mathscr{X}_c) := V(\mathscr{X}_c) = \sum_{x\in\mathscr{X}_c} x - q(x), \tag{5.3}$$

i.e. VLAD aggregates the residuals of the local descriptors for each visual word.

For ASMK, $\sigma(u)$ is defined as a thresholded polynomial selectivity function

$$\sigma(u) := \begin{cases} \text{sign}(u)|u|^\alpha, & \text{if } u > \tau \\ 0, & \text{otherwise,} \end{cases} \tag{5.4}$$

where usually $\alpha = 3$ and $\tau = 0$ and $\Phi(\mathscr{X}_c)$ is chosen as the normalized aggregated residual:

$$\Phi(\mathscr{X}_c) := \hat{V}(\mathscr{X}_c) = V(\mathscr{X}_c) / \left\| V(\mathscr{X}_c) \right\|. \tag{5.5}$$

Regional Search

In this section, we consider image retrieval systems where regional descriptors are stored independently in the database. Denote the query image as X, and the database of N images as $\{Y^{(n)}\}$, $n = 1,2,\ldots,N$. We are mainly interested in the experimental configuration where a query contains a well-localized region-of-interest (i.e. the query in practice contains only one region), which is a common setting in image retrieval. For the n-th database image, regions $r_n = 1,\ldots,R_n$ are predicted by a landmark detector, defining the subimages $\{Y^{(n,r_n)}\}$. We denote $Y^{(n,1)} = Y^{(n)}$ as the subimage corresponding to the original image, and always consider it as a valid region. To leverage uncluttered representations, we store aggregated descriptors independently for each subimage, which leads to a total of $\sum_{n=1}^{N} R_n$ items in the database.

To compute the similarity between the query X and a database image $Y^{(n)}$, we consider max-pooling or average-pooling individual regional similarities, respectively:

$$\text{sim}_{MAX}(X, Y^{(n)}) = max_{r=1,\ldots,R_n} K\left(\mathscr{X}, \mathscr{Y}^{(n,r)}\right) \tag{5.6}$$

$$\text{sim}_{AVG}(X, Y^{(n)}) = \frac{1}{R_n} \sum_{r=1}^{R_n} K\left(\mathscr{X}, \mathscr{Y}^{(n,r)}\right) \tag{5.7}$$

Max-pooling corresponds to assigning a database image's score considering only its highest-scoring subimage. Average pooling aggregates contributions from all subimages. These two variants are compared in section 5.4.

Regional Aggregated Match Kernels

Storing descriptors of each region independently in the database incurs additional cost for both memory and search computation. In this section, we consider utilizing the detected bounding boxes to instead improve the aggregated representations of database images – producing discriminative descriptors at no additional cost. We extend the aggregated match kernel framework of Tolias et al. (2015a) to regional aggregated match kernels, as follows.

We start by noting that the average pooling similarity, Equation (5.7), can be rewritten as:

$$\begin{aligned} &\text{sim}_{AVG}(X, Y^{(n)}) = \\ &\gamma(\mathscr{X}) \sum_c \sum_r \frac{\gamma(\mathscr{Y}^{(n,r)})}{R_n} \sigma\left(\Phi(\mathscr{X}_c)^T \Phi(\mathscr{Y}_c^{(n,r)})\right) \end{aligned} \tag{5.8}$$

Simple regional aggregation. For VLAD, this can be further expanded as:

$$\begin{aligned} &\text{sim}^{(\text{R-VLAD})}(X, Y^{(n)}) \\ &= \gamma(\mathscr{X}) \sum_c \sum_r \frac{\gamma(\mathscr{Y}^{(n,r)})}{R_n} V(\mathscr{X}_c)^T V(\mathscr{Y}_c^{(n,r)}) \\ &= \sum_c \gamma(\mathscr{X}) V(\mathscr{X}_c)^T \sum_r \frac{\gamma(\mathscr{Y}^{(n,r)})}{R_n} V(\mathscr{Y}_c^{(n,r)}) \qquad (5.9) \\ &= \sum_c V_R(\mathscr{X}_c)^T V_R(\{\mathscr{Y}_c^{(n,r)}\}_r) \qquad (5.10) \end{aligned}$$

where we define

$$V_R(\{\mathscr{Y}_c^{(n,r)}\}_r) = \frac{1}{R_n}\sum_r \gamma(\mathscr{Y}^{(n,r)})V(\mathscr{Y}_c^{(n,r)}) \tag{5.11}$$

Using this definition, note that $V_R(\mathscr{X}_c) = \gamma(\mathscr{X})V(\mathscr{X}_c)$. This derivation indicates that average pooling of regional VLAD similarities can be performed using aggregated regional descriptors and does not require storage of each region's representation separately[1]. We refer to this simple regional aggregated kernel as R-VLAD.

A similar derivation can be obtained for ASMK in the case where $\sigma(.)$ is the identity function (i.e. no selectivity is applied), by replacing $V(\mathscr{X}_c)$ by $\hat{V}(\mathscr{X}_c)$ in Equation (5.9). A straightforward matching kernel using this idea would apply the selectivity function when comparing the query ASMK representation against this aggregated representation. We refer to this aggregation variant as Naive-R-ASMK.

Both the R-VLAD and Naive-R-ASMK kernels present an important problem when using many detected regions per image and large codebooks. For a given image region, most visual words will not be associated to any local feature, leading to many all-zero residuals for the region. For visual words that correspond to visual patterns observed in only a small number of regions, this will lead to substantially downweighted residuals. We propose to fix this weakness by developing the R-ASMK kernel as follows, inspired by the changes introduced by the original ASMK with respect to VLAD.

R-ASMK. We define the R-ASMK similarity between a query and a database image as:

$$\begin{aligned}&\text{sim}^{(\text{R-ASMK})}(X^{(n)}, Y^{(n)}) = \\ &\sum_c \sigma\left(\hat{V}_R(\mathscr{X}_c^{(n,r)})^T \hat{V}_R(\{\mathscr{Y}_c^{(n,r)}\}_r)\right)\end{aligned} \tag{5.12}$$

where $\hat{V}_R(\{\mathscr{Y}_c^{(n,r)}\}_r) = \frac{V_R(\{\mathscr{Y}_c^{(n,r)}\}_r)}{\left\|V_R(\{\mathscr{Y}_c^{(n,r)}\}_r)\right\|}$ is the normalized regionally aggregated residual corresponding to visual word c.

R-AMK. The kernels we presented in this section can be regarded as different instantiations of a general regional aggregated match kernel (R-AMK), defined as follows:

[1] Another way to see that this applies to VLAD kernels is to note that VLAD similarity is computed via a simple inner product, and that the average inner product with a set of vectors equals the inner product with the set average; i.e. for vector x and set $\{y_n\}$, $\frac{1}{N}\sum_n x^T y_n = x^T\left(\frac{1}{N}\sum_n y_n\right)$.

$$K_R(X,Y) = \sum_{c \in \mathscr{C}} \sigma\left(\Phi_R(\{\mathscr{X}_c^{(r)}\}_r)^T \Phi_R(\{\mathscr{Y}_c^{(r)}\}_r)\right) \tag{5.13}$$

where $\{\mathscr{X}_c^{(r)}\}_r$ denotes the sets of local descriptors quantized to visual word c, from each region of X. Φ_R specializes to V_R for R-VLAD, and to $\hat{V}_R$ for R-ASMK.

Binarization. For codebooks with a large number of visual words, the storage cost for such aggregated representations can be prohibitive. Binarization is an effective strategy to allow scalable retrieval in these cases. We adopt a similar binarization strategy as proposed by Tolias et al. (2015a), where a binarized version of Φ_R can be obtained by the elementwise function $b(x) = +1$ if $x > 0, -1$ otherwise. We denote the binarized version by a $\star$ superscript (e.g. R-ASMK* is the binarized version of R-ASMK).

5.4 Experiments

We present two types of experiments: first, landmark detection, to assess the quality of object detector models trained on the new dataset. Second, we utilize the detected landmarks to enhance image retrieval systems.

5.4.1 Landmark Detection

We train two types of detection models on the bounding box data that we have collected and described in section 5.3.1: a single shot Mobilenet-V2 (Sandler et al., 2018) based SSD detector (Liu et al., 2016) and a two stage ResNet-50 (He et al., 2016c) based Faster-RCNN (Ren et al., 2015a). Standard object detection evaluation metric Average Precision (AP) measured at 50% intersection-over-union ratio is used during evaluation. Both models reach about 85% AP on the validation set within 500k steps (85.61%, 84.37% respectively). The models are trained with publicly available Tensorflow Object Detection API (Huang et al., 2017b). The results indicate that accurate landmark localization can be trained using our dataset. The Mobilenet-V2-SSD variant runs at 27ms per image, while the ResNet-50-Faster-RCNN runs at 89ms, both numbers on a TitanX GPU.

Learning Curves

We train both Faster-RCNN and SSD based object detection models on our dataset. Figure 5.8 shows the comparison of learning progression of the two models. Both models converge to around 85% mAP within 600k training steps. The MobilenetV2 SSD model trains much

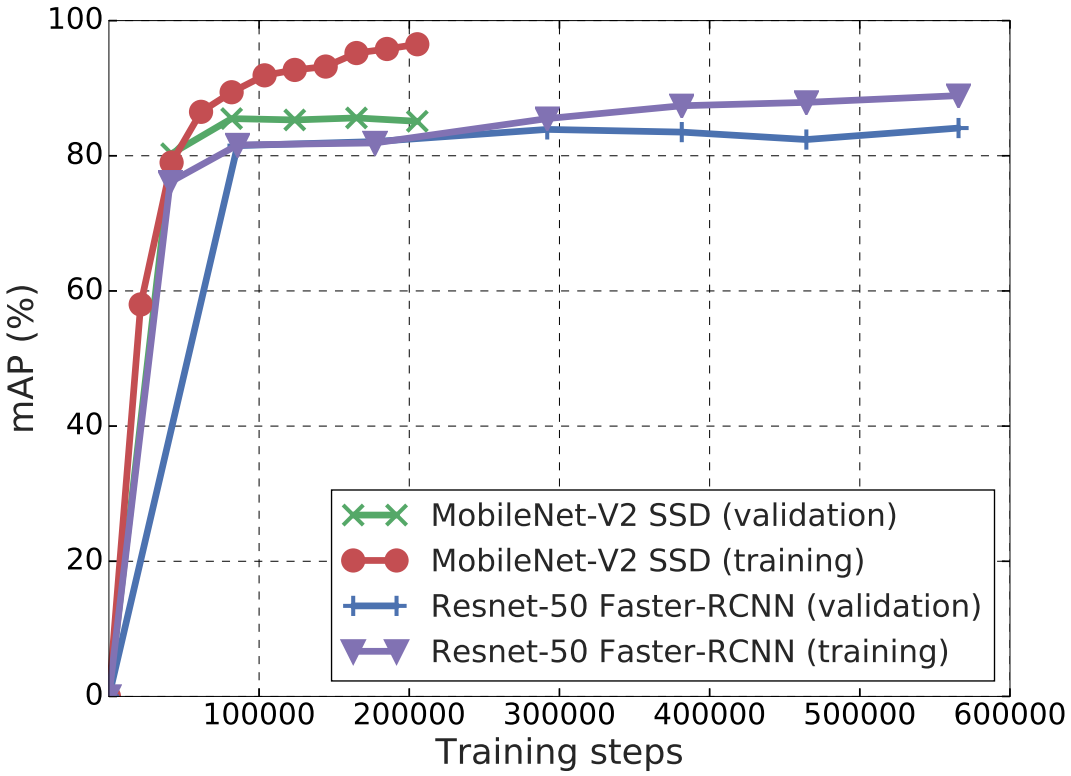


Fig. 5.8 Mean average precision @ IOU=0.5 for the two trained landmark detectors, as a function of the number of training steps.

faster than the ResNet-50 Faster-RCNN, due to much smaller model size and larger batch size (32 vs. 1, respectively). We also observe that SSD-based model slightly outperforms the Faster-RCNN base model despite having a smaller/weaker feature extractor. We conjecture that the advantage is due to the multi-scale feature map of SSD capturing the landmarks at different scales better than Faster-RCNN, which operates on a single feature map.

Qualitative evaluation of the detector

To illustrate the effectiveness of our trained detectors, we present examples of detection using the SSD model. Figure 5.9 shows examples for a variety of landmarks with different scales, occlusion and lighting conditions. In addition, we also show some failure cases in Figure 5.10 where the object of interest has ambiguous semantic boundary (resulting in double-detection) or is very hard to distinguish from the scene (resulting in missed detection). For both figures, only detections with confidence probability more than 0.2 are shown.

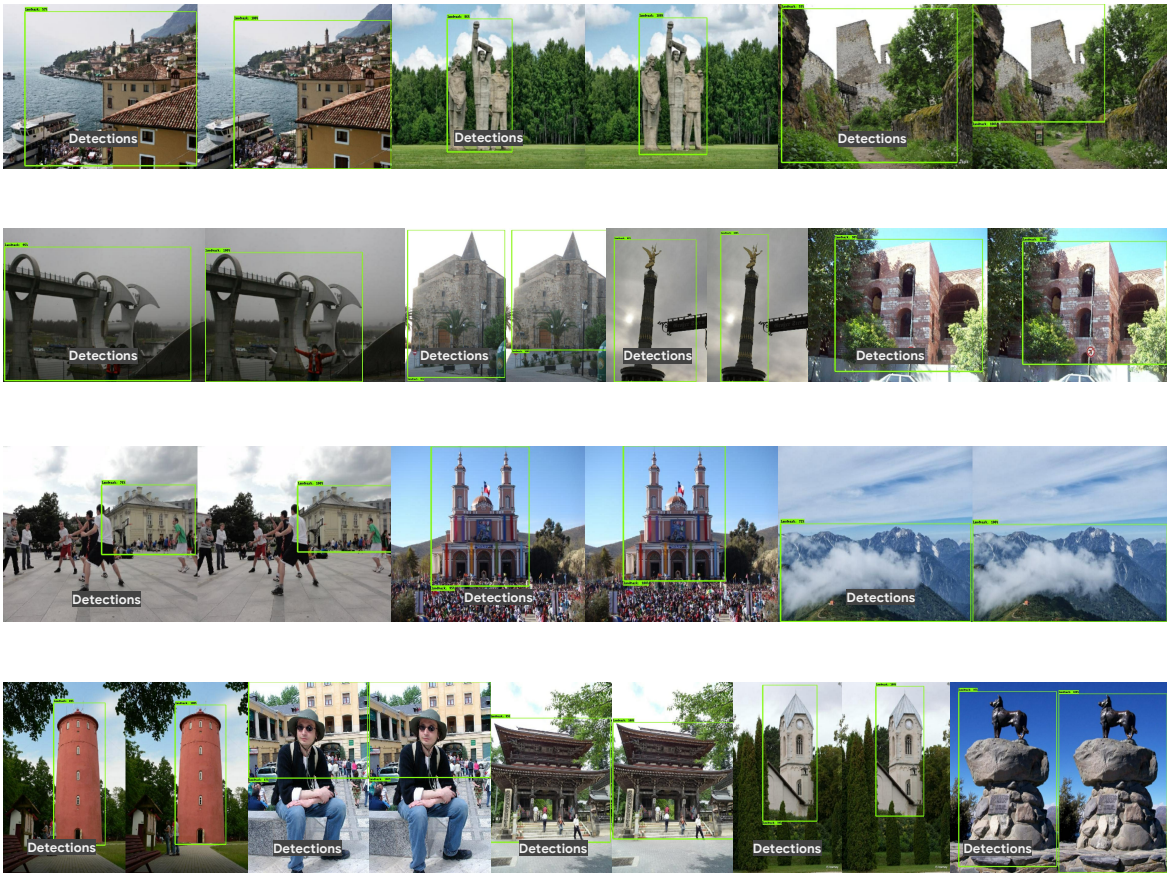


Fig. 5.9 Detection (on the left) versus ground truth (on the right) on the Google Landmarks dataset.

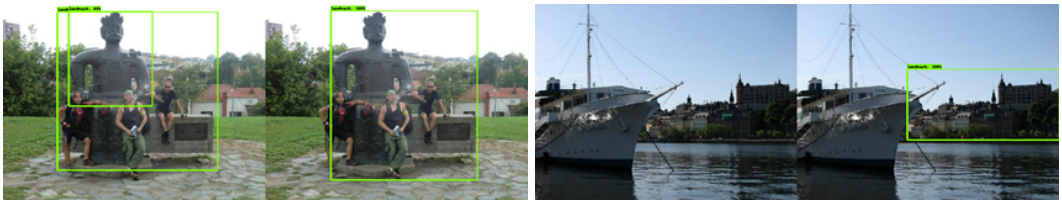


Fig. 5.10 Two failure detection cases. On the right are the ground truth images, and on the left are the outputs of the detector (if any).

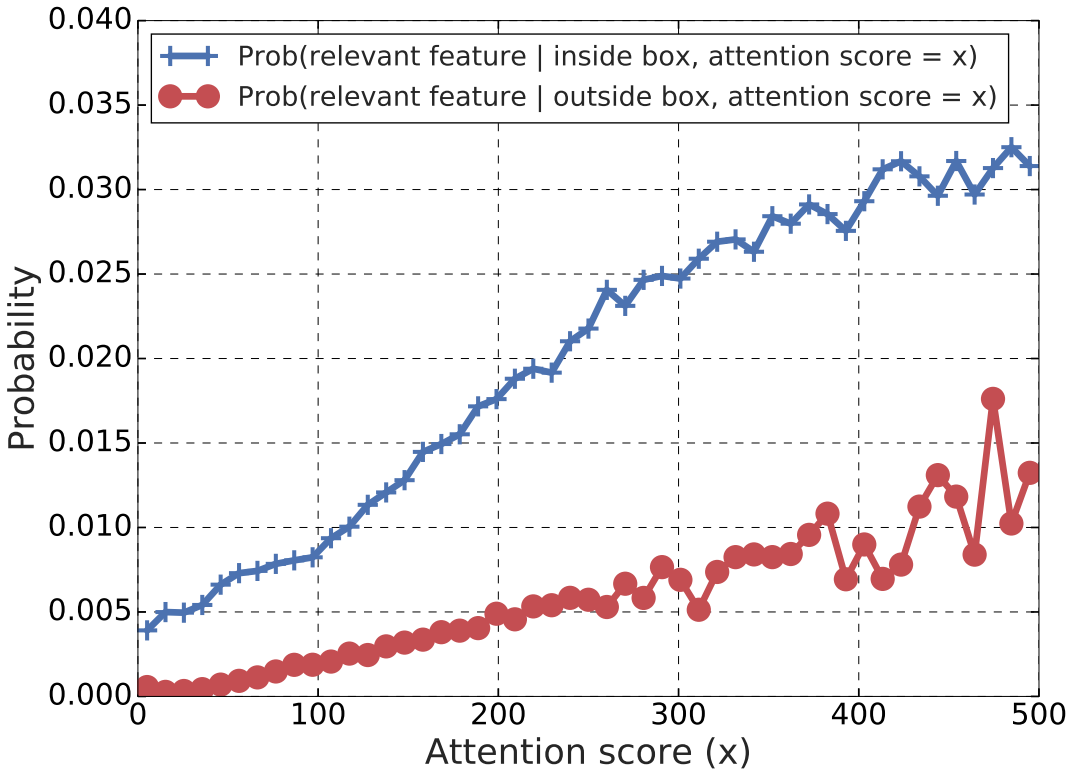


Fig. 5.11 Relevance probability of a DELF local feature, as a function of its attention score. The blue curve denotes features inside predicted bounding boxes, while the red curve denotes features outside them. The detected boxes provide valuable information that can be used to improve image representations for retrieval tasks.

Relevants of detected features

In this section, we analyse the detector's ability to focus on relevant landmarks by empirically estimating the proportion of relevant local features located within or without predicted bounding boxes. We extract and match local features for image pairs that are known to depict the same landmark. A local feature is declared to be relevant if it is an inlier to a high-confidence estimated geometric transformation.

More specifically, we use DELF local features (Noh et al., 2017c) and a Faster-RCNN (Ren et al., 2015a) landmark detector trained on our new dataset. 10k image pairs are collected from the Google Landmarks dataset (Noh et al., 2017c). Local feature matching is performed via nearest neighbour search followed by geometric verification (RANSAC (Fischler and Bolles, 1981a) with an affine model). Figure 5.11 plots the relevance probabilities as a function of the DELF local feature attention scores (these attention scores can be interpreted

as a measure of a local feature's "landmarkness"). The blue curve denotes features that are located within bounding boxes, while the red curve represents features located outside bounding boxes.

The curves show that local features located within bounding boxes are much more likely to be relevant: for two features with the same attention score, the relevance probability for a feature located within a predicted box is approximately 3 to 4$\times$ larger than that for a local feature located outside the box. Note how feature relevance increases with attention scores, as expected, but the predicted boxes can provide important extra information to effectively select the best features. This can be interpreted as the merging of two information streams: *bottom-up* (DELF attention scores estimate per-local feature relevance) and *top-down* (landmark detector estimates relevance of large regions).

Our proposed R-ASMK* can be seen as a local feature re-weighting mechanism, which favors features located within detected regions. The experimental results obtained on the $\mathscr{R}$Oxford and $\mathscr{R}$Paris datasets confirm that re-weighting features within detected regions boost image retrieval performance substantially.

Overall this experiment confirms that detection helps identifying relevant local features.

5.4.2 Image Retrieval

We perform regional search and regional aggregation experiments. The following describes the experimental setup.

Datasets. We use the Oxford (Philbin et al., 2007) and Paris (Philbin et al., 2008b) datasets, which have recently been revisited to correct annotation mistakes, add new query images and introduce new evaluation protocols (Radenović et al., 2018); the datasets are referred to as $\mathscr{R}$Oxf and $\mathscr{R}$Par, respectively. There are 70 query images for each dataset, with 4993 (6322) database images in the $\mathscr{R}$Oxf ($\mathscr{R}$Par) dataset. We report results on the Medium and Hard setups; for ablations, we focus more specifically on the Hard setup. Performance is measured using mean average precision (mAP) and mean precision at rank 10 (mP@10). We also perform large-scale experiments using the $\mathscr{R}$1M distractor set introduced by Radenović et al. (2018), which contains 1,001,001 images.

Image representation. We use the following setup in our experiments, except where indicated otherwise. The released DELF model (Noh et al., 2017c) (pre-trained on the dataset from Gordo et al. (2016)) is used, with the default configuration (maximum of 1000 features per region are extracted, with a required minimum attention score of 100), except that the feature dimensionality is set to 128 as in previous work (Radenović et al., 2018).

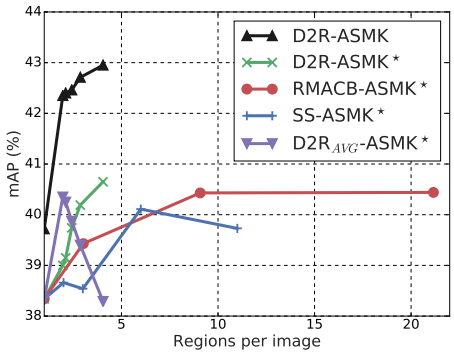
A 1024-sized codebook is used when computing aggregated kernels; as common practice, codebooks are trained on $\mathscr{R}$Oxf for retrieval experiments on $\mathscr{R}$Par, and vice versa. We focus on improving the core image representations for retrieval, and do not consider query expansion (QE) (Chum et al., 2007) techniques such as Hamming QE (Tolias and Jegou, 2014), α QE (Radenović et al., 2018) or diffusion (Iscen et al., 2018, 2017); these methods could be incorporated into our system to obtain even stronger retrieval performance.

Region selection techniques. For our Detect-to-Retrieve (D2R) framework, we adopt the trained Faster RCNN detector described in section 5.4.1. We compare against previously proposed region selection techniques for image retrieval: the uniform grid from Razavian et al. (2016); Tolias et al. (2015b) (denoted RMACB, for "RMAC boxes") and Selective Search (SS) (Tao et al., 2014; Uijlings et al., 2013). To vary the number of regions per image, we do as follows: (i) for D2R, we vary the landmark detector threshold; (ii) for RMACB, we sweep the number of levels from 1 to 3; (iii) for SS, we select the top $\{1, 2, 5, 10\}$ boxes per image (as in this case there are no confidence scores associated to regions). For all region selection techniques, we add the original image as one of the selected regions.

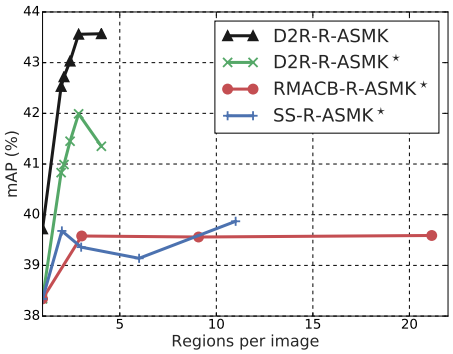
Implementation details. We implemented the aggregated kernel framework from scratch in Python/Tensorflow. As a comparison against the reference MATLAB implementation given by Tolias et al. (2015a), our ASMK* with a 1024-sized codebook and DELF features obtains 37.91% mAP in the $\mathscr{R}$Oxf-Hard dataset, while the reference implementation obtains 37.08%. Note that the reference implementation uses a similar configuration as Hamming Embedding (HE) (Jégou et al., 2008), with a projection matrix before binarization, residuals computed with respect to the median, and IDF. We did not find consistent improvements using these, so we use the simpler version as described in section 5.3.2. Similarly, the reference implementation uses multiple visual word assignments, but our preliminary experiments show improved results using single assignment, making retrieval faster and simpler – therefore we adopt single assignment in our experiments. We extended this implementation to support our regional search and aggregation techniques.

Regional Search

We compare aggregated match kernels, region selection techniques and similarity computation methods on the $\mathscr{R}$Oxf-Hard dataset. When performing regional search, multiple regions are selected per image and stored independently in the database, leading to increased memory cost. Figure 5.12(a) presents results for ASMK variants, where all techniques use max-pooling similarity from Equation (5.6), except for D2R$_{AVG}$-ASMK*, which uses



(a) Regional search evaluation.



(b) Regional aggregation evaluation.

Fig. 5.12 Regional search and aggregation evaluations of different image representations, on $\mathscr{R}$Oxf-Hard. (a) Regional search: each regional representation is stored independently in the database, leading to increased memory requirements. Our D2R-ASMK variants achieve significant improvements over the single-image baseline while requiring substantially fewer boxes compared to other region selection approaches. (b) Regional aggregation: each region contributes to the aggregated representation for the entire image. The aggregated descriptor dimensionality is identical to a single-image baseline that does not use regions. Our D2R-R-ASMK variants leverage the different landmark regions to compose a strong image representation, which is even more effective than storing each regional representation separately.

Method	Det. Thresh.	$\mathscr{R}$Oxf-Hard mAP	$\mathscr{R}$Oxf-Hard Size	$\mathscr{R}$Par-Hard mAP	$\mathscr{R}$Par-Hard Size
ASMK*	—	38.3	1	54.2	1
D2R-ASMK*	0.7	39.2	2.1	56.0	2.2
	0.5	39.7	2.4	56.2	2.4
	0.3	40.2	2.9	56.3	2.9
	0.1	40.7	4.1	56.7	3.9
D2R-R-ASMK*	0.7	41.0	1	56.2	1
	0.5	41.5	1	56.2	1
	0.3	**42.0**	1	56.3	1
	0.1	41.4	1	**56.8**	1

Table 5.2 Retrieval mAP and relative database size for the different region-based techniques introduced in this work, on the $\mathscr{R}$Oxf-Hard and $\mathscr{R}$Par-Hard datasets, as a function of the landmark detector threshold used for region selection. D2R-ASMK* uses max-pooling similarity from Equation (5.6). The performances of both D2R-ASMK* and D2R-R-ASMK* tend to improve as the detection threshold decreases (more regions are selected). D2R-R-ASMK* outperforms D2R-ASMK* consistently, with a smaller memory footprint.

average-pooling similarity from Equation (5.7). Combining our proposed D2R regions with ASMK enhances mAP by 3.23% when using an average of 4.05 regions per image.

We compare the different region selection approaches using ASMK*. Our D2R-ASMK* achieves 40.65% mAP when using 4.05 regions per image, an improvement of 2.31% over the single-image ASMK* baseline. Other region selection approaches improve retrieval accuracy, but with significantly larger memory requirements. RMACB-ASMK* requires 9.08 regions/image to achieve 40.43% mAP (this is 0.22% mAP below the previously mentioned D2R-ASMK* operating point, despite requiring 2.24$\times$ the memory). SS-ASMK* benefits from some regions, while performance decreases when a large number of regions are selected, since many of those regions are irrelevant.

Average pooling of individual regional similarities improves upon the single-image baseline significantly, at low overhead memory requirements: D2R$_{AVG}$-ASMK* achieves 40.35% mAP with only 1.96$\times$ storage cost. Note that in this case performance drops significantly as more regions are added, since irrelevant regional similarities are added to the final image similarity. We also experimented with a D2R-VLAD representation: mAP improves from 30.17% (single-image) to 33.87% (2.87 regions/image).

Table 5.2 further presents D2R-ASMK* results on the $\mathscr{R}$Par-Hard dataset. Regional search enables 2.5% mAP improvement at 3.9 regions/image. Note that our D2R approach

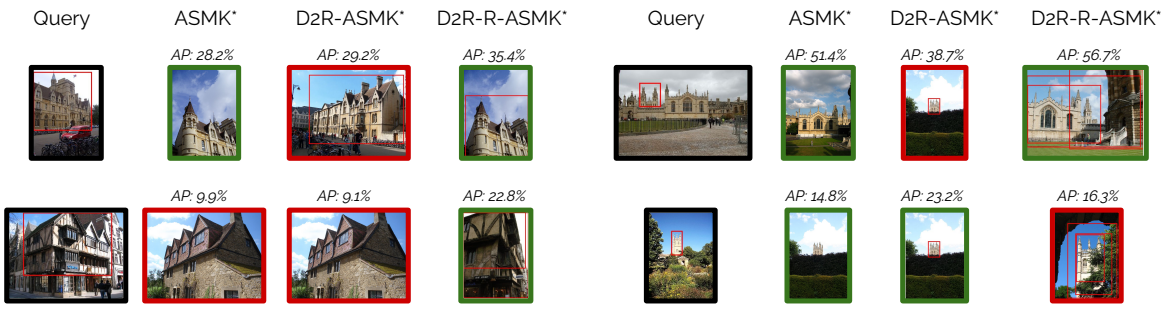


Fig. 5.13 Qualitative results for ASMK⋆ (baseline single-image method), D2R-ASMK⋆ (regional search) and D2R-R-ASMK⋆ (regional aggregation) on $\mathscr{R}$Oxf-Hard. Four queries are presented, with their regions-of-interest highlighted. For each method, we show the first ranked image where the methods disagree. Red borders indicate incorrect results, and green borders indicate correct results. For D2R-ASMK⋆, we box the region used for the result (or leave unboxed if the region corresponds to the entire image). For D2R-R-ASMK⋆, we box all regions used for aggregation. We also present average precision (AP) for each method and query.

is effective even if the landmarks in the Google Landmark Boxes dataset present much larger variability than the landmarks encountered in the $\mathscr{R}$Oxf/$\mathscr{R}$Par datasets.

Regional Aggregated Match Kernels

In this section, we evaluate the proposed regional aggregated match kernels. In this experiment, region selection is used to produce an improved image representation, with no increase in the aggregated descriptor dimensionality. Figure 5.12(b) compares different aggregation methods and region selection approaches, on the $\mathscr{R}$Oxf-Hard dataset. Both our proposed D2R-R-ASMK and D2R-R-ASMK⋆ variants achieve substantial improvements compared to their baselines which do not use boxes for aggregation: 3.85% and 3.65% absolute mAP improvements, respectively. We also compare our D2R approach against other region selection methods. RMACB and SS improve upon the baseline, but with limited gain of at most 1.5% mAP.

More interestingly, our proposed kernels outperform even the regional search configuration where each region is indexed separately in the database. Table 5.2 compiles experimental results on $\mathscr{R}$Oxf-Hard and $\mathscr{R}$Par-Hard. Our D2R-R-ASMK⋆ method outperforms the best regional search variant on both datasets, respectively by 1.3% and 0.1% absolute mAP, with relative storage savings of 4.1× and 3.9×.

In another ablation experiment, we assess the performance of simpler regional aggregation methods: R-VLAD and Naive-R-ASMK. We use the trained detector to select regions. For

Method	Medium							
	$\mathscr{R}$Oxf		$\mathscr{R}$Oxf+$\mathscr{R}$1M		$\mathscr{R}$Par		$\mathscr{R}$Par+$\mathscr{R}$1M	
	mAP	mP@10	mAP	mP@10	mAP	mP@10	mAP	mP@10
AlexNet-GeM (Radenović et al., 2018)	43.3	62.1	24.2	42.8	58.0	91.6	29.9	84.6
VGG16-GeM (Radenović et al., 2018)	61.9	82.7	42.6	68.1	69.3	97.9	45.4	94.1
ResNet101-R-MAC (Gordo et al., 2016)	60.9	78.1	39.3	62.1	78.9	96.9	54.8	93.9
ResNet101-GeM (Radenović et al., 2018)	64.7	84.7	45.2	71.7	77.2	98.1	52.3	95.3
ResNet101-GeM↑+DSM (Siméoni et al., 2019)	65.3	87.1	47.6	76.4	77.4	99.1	52.8	96.7
HesAff-rSIFT-ASMK⋆ (Tolias et al., 2015a)	60.4	85.6	45.0	76.0	61.2	97.9	42.0	95.3
HesAff-rSIFT-ASMK⋆+SP (Tolias et al., 2015a)	60.6	86.1	46.8	79.6	61.4	97.9	42.3	95.3
HesAff-HardNet-ASMK⋆+SP (Mishkin et al., 2018)	65.6	90.2	–	–	65.2	98.9	–	–
DELF-ASMK⋆+SP (Noh et al., 2017c; Radenović et al., 2018)	67.8	87.9	53.8	81.1	76.9	99.3	57.3	98.3
DELF-ASMK⋆ (reimpl.)	65.7	87.9	–	–	77.1	98.7	–	–
DELF-D2R-R-ASMK⋆ (ours)	69.9	89.0	–	–	78.7	99.0	–	–
— DELF-GLD (ours)	73.3	90.0	61.0	84.6	**80.7**	99.1	**60.2**	97.9
DELF-ASMK⋆+SP (reimpl.)	68.9	90.9	–	–	76.6	98.7	–	–
DELF-D2R-R-ASMK⋆+SP (ours)	71.9	91.3	–	–	78.0	**99.4**	–	–
— DELF-GLD (ours)	**76.0**	**93.4**	**64.0**	**87.7**	80.2	99.1	59.7	**99.0**

Method	Hard							
	$\mathscr{R}$Oxf		$\mathscr{R}$Oxf+$\mathscr{R}$1M		$\mathscr{R}$Par		$\mathscr{R}$Par+$\mathscr{R}$1M	
	mAP	mP@10	mAP	mP@10	mAP	mP@10	mAP	mP@10
AlexNet-GeM (Radenović et al., 2018)	17.1	26.2	9.4	11.9	29.7	67.6	8.4	39.6
VGG16-GeM (Radenović et al., 2018)	33.7	51.0	19.0	29.4	44.3	83.7	19.1	64.9
ResNet101-R-MAC (Gordo et al., 2016)	32.4	50.0	12.5	24.9	59.4	86.1	28.0	70.0
ResNet101-GeM (Radenović et al., 2018)	38.5	53.0	19.9	34.9	56.3	89.1	24.7	73.3
ResNet101-GeM↑+DSM (Siméoni et al., 2019)	39.2	55.3	23.2	37.9	56.2	89.9	25.0	74.6
HesAff-rSIFT-ASMK⋆ (Tolias et al., 2015a)	36.4	56.7	25.7	42.1	34.5	80.6	16.5	63.4
HesAff-rSIFT-ASMK⋆+SP (Tolias et al., 2015a)	36.7	57.0	26.9	45.3	35.0	81.7	16.8	65.3
HesAff-HardNet-ASMK⋆+SP (Mishkin et al., 2018)	41.1	59.7	–	–	38.5	87.9	–	–
DELF-ASMK⋆+SP (Noh et al., 2017c; Radenović et al., 2018)	43.1	62.4	31.2	50.7	55.4	**93.4**	26.4	75.7
DELF-ASMK⋆ (reimpl.)	41.0	57.9	–	–	54.6	90.9	–	–
DELF-D2R-R-ASMK⋆ (ours)	45.6	61.9	–	–	57.7	93.0	–	–
— DELF-GLD (ours)	47.6	64.3	33.6	53.7	**61.3**	**93.4**	**29.9**	82.4
DELF-ASMK⋆+SP (reimpl.)	46.6	66.7	–	–	52.2	87.6	–	–
DELF-D2R-R-ASMK⋆+SP (ours)	48.5	66.7	–	–	54.0	87.6	–	–
— DELF-GLD (ours)	**52.4**	**70.9**	**38.1**	**61.3**	58.6	91.0	29.4	**83.9**

Table 5.3 Comparison of proposed techniques against state-of-the-art methods, on the $\mathscr{R}$Oxford ($\mathscr{R}$Oxf) and $\mathscr{R}$Paris ($\mathscr{R}$Par) datasets (and their large-scale extensions $\mathscr{R}$Oxf+$\mathscr{R}$1M and $\mathscr{R}$Par+$\mathscr{R}$1M), with medium and hard evaluation protocols. Previously published results are presented in the first block of rows. The second and third block of rows present our experimental results, considering systems without and with spatial verification (SP), respectively. In this experiment, we use codebooks with 65k visual words, to make our results comparable to the work of Radenović et al. (2018). DELF-GLD indicates a version of DELF which we re-trained on the Google Landmarks dataset. Our methods achieve equal or improved performance for all evaluation protocols, datasets and metrics.

R-VLAD, mAP on $\mathscr{R}$Oxf improves from 30.17% (single-image) to 30.91% when using 2.4 regions per image, but degrades quickly as more regions are considered. In particular, when setting a very low detection threshold (0.01) to obtain 10.2 regions per image, performance degenerates to 16.46% mAP – this agrees with the intuition that a large number of regions is detrimental to R-VLAD. For Naive-R-ASMK, no improvement is obtained when detected regions are used: mAP drops from 39.72% to 31.42% when 1.96 regions per image are used, and similarly degenerates to 9.2% when using 10.2 regions per image. In comparison, using the same detection threshold of 0.01, R-ASMK* obtains 41.6% mAP, i.e. performance is high even if using a large number of regions, due to the improved aggregation technique.

Comparison Against State-of-the-Art

We compare our D2R-R-ASMK* technique against state-of-the-art image retrieval systems. To make our system comparable with the results published by Radenović et al. (2018), for this experiment we use a codebook with 65k visual words. We also further experiment with re-training the DELF local feature on the Google Landmarks dataset (denoted as DELF-GLD). Spatial verification (SP) is used to re-rank the top 100 database images (we use RANSAC with an affine model).

Table 5.3 presents experimental results on $\mathscr{R}$Oxf and $\mathscr{R}$Par, using the medium and hard protocols, also including the large-scale setup with $\mathscr{R}$1M. Our proposed D2R-R-ASMK* representation by itself, without spatial verification, already improves mAP when comparing against all previously published results. SP further boosts performance by about 3% mAP on $\mathscr{R}$Oxf; surprisingly, it slightly degrades performance on the $\mathscr{R}$Par dataset. Re-training DELF on GLD improves performance by around 4%. Our best results improve upon the previous state-of-the-art by 8.2% mAP on $\mathscr{R}$Oxf-Medium, 1.8% mAP on $\mathscr{R}$Par-Medium, 9.3% mAP on $\mathscr{R}$Oxf-Hard and 1.9% in $\mathscr{R}$Par-Hard (with similar gains in the large-scale setup).

Memory. Our DELF-D2R-R-ASMK* descriptors have the exact same dimensionality as DELF-ASMK*. However, DELF-ASMK* is sparser and consumes less memory in practice: 10.3GB, compared to 27.6GB for DELF-D2R-R-ASMK*, in the large-scale $\mathscr{R}$Oxf+$\mathscr{R}$1M dataset. This is still much less than other local feature based approaches: e.g. HesAff-rSIFT-ASMK* requires 62GB (Radenović et al., 2018) and HesAffNet-HardNet++-ASMK* (Mishkin et al., 2018) requires approximately 86.8GB.

Qualitative Region Selection Comparison

In this section, we present landmark detection results on the $\mathscr{R}$Oxford and $\mathscr{R}$Paris datasets (Figure 5.14 and Figure 5.15, respectively), comparing with the selected regions by competitive approaches (RMAC boxes and Selective Search). The three methods use a configuration that produces a roughly similar number of regions per image: D2R with detection threshold 0.1 (about 4 regions per image), RMAC boxes with 2 levels (9 regions per image), and Selective Search with 6 selected regions per image. Note that our image retrieval experiments always use the whole image as a valid region, but in these visualizations we do not box the whole image, for a more concise presentation.

The figures show that our trained landmark detector tends to focus on the most prominent landmark regions in the image. RMAC boxes correspond to a fixed multi-scale grid, where the selected regions only depend on the input image size, not on its contents. This leads to regularly spaced boxes which do not usually overlap well with landmarks. Selective search produces boxes corresponding to prominent objects in the scene, which may or may not correspond to landmarks.

Discussion

Our experiments demonstrate that selecting relevant image regions can help boost image retrieval performance significantly. In our regional aggregation method, the detected regions allow for effective re-weighting of local feature contributions, emphasizing relevant visual patterns in the final image representation. Note, however, that it is crucial to perform **both** region selection **and** regional aggregation in a suitable manner. If the selected regions are not relevant to the objects of interest, regional aggregation cannot be very effective, as shown in Figure 5.12(b). Also, our experiments with naive versions of regional aggregation indicate that the aggregation needs to be performed in the right way: this is related to the poor R-VLAD and Naive-R-ASMK results.

It may initially seem unintuitive that the regional search method underperforms when compared to our regional aggregation technique. However, this can be understood by observing some retrieval result patterns, which are presented in Figure 5.13. The addition of separate regional representations to the database may help retrieval of relevant small objects in cluttered scenes, as illustrated with the successful bottom-right D2R-ASMK* retrieved image. However, it also increases the chances of finding localized regions which are similar but do not correspond to the same landmark, as illustrated with the top two cases.

Regional aggregation, on the other hand, can help retrieval by re-balancing the visual information presented in an image. The top-right D2R-R-ASMK* result shows a database

Fig. 5.14 Examples of selected regions for the three methods compared in the paper, on the $\mathscr{R}$Oxford dataset. Left: our D2R approach, with detection threshold of 0.1 (4.1 regions per image). Centre: RMAC boxes (fixed multi-scale grid), with 2 levels (9 regions per image). Right: Selective search, with 6 regions per image. Note that edges for some regions overlap in some cases, so not all regions may be clearly visible.

Fig. 5.15 Examples of selected regions for the three methods compared in the paper, on the $\mathscr{R}$Paris dataset. Left: our D2R approach, with detection threshold of 0.1 (3.9 regions per image). Centre: RMAC boxes (fixed multi-scale grid), with 2 levels (9 regions per image). Right: Selective search, with 6 regions per image. Note that edges for some regions overlap in some cases, so not all regions may be clearly visible.

image where the detected boxes do not precisely cover the query object; instead, several selected regions cover it, and consequently its features are boosted. A similar case is illustrated in the bottom-left example, where the main detected region in the database image does not cover the object of interest entirely. The features inside the main box are boosted but those outside are also used, generating a more suitable representation for image retrieval.

5.5 Conclusion

In this chapter we have seen how to improve upon the state-of-the-art in image retrieval by combining the global knowledge of a detection system with the DELF local feature descriptor. In order to achieve this improvement we use the structure of the underlying task. This allows us to improve not only the performance but also the storage efficiency and the inference speed of the underlying system. This chapter provides another good example of how we improve the performance of one task by integrating knowledge that we have obtained from solving another task.

Limitations & Further Work: Our system is able to beat the state of the art in landmark recognition. It works incredibly well and only a few failure cases could be observed and most of these involve a failed detection prediction. The simplest way to improve detection performance would be to additionally label images from the Oxford and Paris dataset with detection ground truth. Another direction would be to try to improve the detector by using attention gradients. Currently the relationship between detection and attention is only used at inference time. By also applying the detection weights during attention training the detection could be improved, even on examples that do not provide detection ground truth.

Chapter 6

Conclusion

In this thesis we have shown that structured modelling is a very strong alternative to end-to-end learning. We have started by discussing some of the limitations of end-to-end training. We have shown several examples where we were able to overcome those limitations by using structured modelling. We have shown that using an intermediate representation helps to improve generalization, interpretability and robustness and this has enabled us to achieve state-of-the-art performance.

Building a geometric intermediate representation is particularly useful for autonomous driving and robotics applications. Visual perception in such applications can be considered to be a multi-modal problem. We have shown that we can use this to improve computational efficiency by sharing features in an encoder network between tasks. By explicitly creating an intermediate representation we can go even further and use the structural relationships between tasks to improve performance.

6.1 Limitations & Future Work

While structured modelling is a powerful tool that should be used in appropriate situations, it is not a silver bullet. It comes with limitations and much more research needs to be done to further improve multi-modal visual perception in the context of autonomous driving. We have discussed specific limitations and possible improvements for the individual contributions in the body of this thesis. In this section we add some more general directions in which we think further research could be beneficial.

6.1.1 Obtain theoretical guarantees

A common criticism of deep neural networks is that their outputs are hard to explain. Neural networks are often described as a black-box predictor. While we can experimentally verify whether the network behaves as expected in a predefined setting we don't have many theoretical guarantees about the behaviour in uncommon situations. In addition, it is well known that deep learning models can behave very unreasonably when the input is too far away from the manifold of the original training data (Goodfellow et al., 2014).

This is a major issue for robotics and for autonomous driving applications. For safety reasons, it would be desirable to have some guarantee or understanding that we can trust the output of our perception network. Uncertainty (Kendall and Gal, 2017) is one approach suggested for evaluating the strength of a prediction. An alternative would be to apply geometric reasoning to the intermediate representations in order to assess the trustworthiness of a prediction.

6.1.2 Including temporal information

In most autonomous driving applications the sensor is able to provide a stream of images as videos. One very interesting area for future research would be to take advantage of the temporal information provided by a video. One option would be to compute one geometric representation for the entire video. For real-time applications, it might be feasible to have one representation that is constantly updated. This would allow us to exploit video data to gain temporal consistency and improve prediction performance.

6.1.3 Applying structured modelling to more tasks

Finally, we see a lot of potential for applying the approach of structured modelling to many more tasks. Using intermediate representation could be particularly useful for tasks such as tracking, 3D detection and predicting future outcomes. One very interesting research question would be to find a single representation that could be used for solving many of those tasks. This also leads to interesting trade-off questions between the complexity of the representation and its modelling potential.

References

(2019). OpenSFM structure from motion library. https://github.com/mapillary/OpenSfM. Accessed: 2019-03-21.

Adams, A., Baek, J., and Davis, M. A. (2010). Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library.

Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. In *Proc. CVPR*.

Arandjelović, R. and Zisserman, A. (2012). Three things everyone should know to improve object retrieval. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918. IEEE.

Arandjelovic, R. and Zisserman, A. (2013). All About VLAD. In *Proc. CVPR*.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965.

Bohyung Han, Andre Araujo, B. C. S.-F. C. O. C. T. S. J. S. G. T. T. W. X. Z. (2019). Google landmarks dataset. https://www.kaggle.com/c/landmark-recognition-2019.

Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., and Rother, C. (2017). DSAC - differentiable RANSAC for camera localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Brachmann, E. and Rother, C. (June 2018). Learning less is more - 6D camera localization via 3d surface regression. In *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Brostow, G., Fauqueur, J., and Cipolla, R. (2009). Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97.

Buddemeier, U. and Neven, H. (2012). Systems and Methods for Descriptor Vector Computation. US Patent 8,098,938.

Budvytis, I., Sauer, P., and Cipolla, R. (September 2018). Semantic localisation via globally unique instance segmentation. In *British Machine Vision Conference (BMVC)*.

Budvytis, I., Sauer, P., Roddick, T., Breen, K., and Cipolla, R. (2017). Large scale labelled video data augmentation for semantic segmentation in driving scenarios. In *5th Workshop on Computer Vision for Road Scene Understanding and Autonomous Driving in IEEE International Conference on Computer Vision (ICCV)*.

Budvytis*, I., Teichmann*, M., Vojir*, T., and Cipolla, R. (2019). Large scale joint semantic re-localisation and scene understanding via globally unique instance coordinate regression. *British Machine Vision Conference (BMVC)*.

Caltagirone, L., Scheidegger, S., Svensson, L., and Wahde, M. (2017). Fast lidar-based road detection using convolutional neural networks. *arXiv preprint arXiv:1703.03613*.

Chandra, S. and Kokkinos, I. (2016). Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In *European Conference on Computer Vision*, pages 402–418. Springer.

Chen, D. M., Baatz, G., Köser, K., Tsai, S. S., Vedantham, R., Pylvänäinen, T., Roimela, K., Chen, X., Bach, J., Pollefeys, M., Girod, B., and Grzeszczuk, R. (2011). City-scale landmark identification on mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062.

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2015a). Semantic image segmentation with deep convolutional nets and fully connected crfs. *International Conference on Learning Representations*.

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2016a). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017a). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., and Urtasun, R. (2016b). Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156.

Chen, X., Kundu, K., Zhu, Y., Berneshawi, A. G., Ma, H., Fidler, S., and Urtasun, R. (2015b). 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432.

Chen, Y.-h., Lopez-Moreno, I., Sainath, T. N., Visontai, M., Alvarez, R., and Parada, C. (2015c). Locally-connected and convolutional neural networks for small footprint speaker recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. (2017b). Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257*.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning. *CoRR*, abs/1410.0759.

Chum, O., Philbin, J., Sivic, J., Isard, M., and Zisserman, A. (2007). Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval. In *Proc. ICCV*.

Cohen, A., Schönberger, J. L., Speciale, P., Sattler, T., Frahm, J. M., and Pollefeys, M. (2016). Indoor-outdoor 3D reconstruction alignment. In *European Conference on Computer Vision (ECCV)*.

Dai, J., He, K., and Sun, J. (2016). Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009a). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009b). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2013). Scalable object detection using deep neural networks. *CoRR*, abs/1312.2249.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2012). Pascal voc dataset. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Fischler, M. and Bolles, R. (1981a). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395.

Fischler, M. A. and Bolles, R. C. (1981b). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

Fritsch, J., Kuehnl, T., and Geiger, A. (2013). A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*.

Gao, H., Yuan, H., Wang, Z., and Ji, S. (2017). Pixel deconvolutional networks. *CoRR*, abs/1705.06820.

Geiger, A. (2013). Kitti road public benchmark.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gidaris, S. and Komodakis, N. (2015). Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1134–1142.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524.

Giusti, A., Ciresan, D. C., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2013). Fast image scanning with deep max-pooling convolutional neural networks. *CoRR*, abs/1302.1700.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Gordo, A., Almazan, J., Revaud, J., and Larlus, D. (2016). Deep Image Retrieval: Learning Global Representations for Image Search. In *Proc. ECCV*.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., and Malik, J. (2011). Semantic contours from inverse detectors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 991–998. IEEE.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2014). Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017a). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017b). Mask R-CNN. *CoRR*, abs/1703.06870.

He, K., Zhang, X., Ren, S., and Sun, J. (2015a). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

He, K., Zhang, X., Ren, S., and Sun, J. (2015b). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016c). Deep Residual Learning for Image Recognition. In *Proc. CVPR*.

He, K., Zhang, X., Ren, S., and Sun, J. (June 2016b). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

He, X. and Zemel, R. S. (2009). Learning hybrid models for image annotation with partially labeled data. In *Advances in Neural Information Processing Systems*, pages 625–632.

Hosang, J. H., Benenson, R., Dollár, P., and Schiele, B. (2015). What makes for effective detection proposals? *CoRR*, abs/1502.05082.

Hosang, J. H., Benenson, R., and Schiele, B. (2014). How good are detection proposals, really? *CoRR*, abs/1406.6962.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017a). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al. (2017b). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311.

Im, D. J., Kim, C. D., Jiang, H., and Memisevic, R. (2016). Generating images with recurrent adversarial networks. *CoRR*, abs/1602.05110.

Iscen, A., Avrithis, Y., Tolias, G., Furon, T., and Chum, O. (2018). Fast Spectral Ranking for Similarity Search. In *Proc. CVPR*.

Iscen, A., Tolias, G., Avrithis, Y., Furon, T., and Chum, O. (2017). Efficient Diffusion on Region Manifolds: Recovering Small Objects with Compact CNN Representations. In *Proc. CVPR*.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., and Fitzgibbon, A. (2011). Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User interface Software and Technology*.

Jégou, H., Douze, M., and Schmid, C. (2008). Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Proc. ECCV*.

Jégou, H., Douze, M., Schmidt, C., and Perez, P. (2010). Aggregating Local Descriptors into a Compact Image Representation. In *Proc. CVPR*.

Jégou, H., Perronnin, F., Douze, M., Sanchez, J., Perez, P., and Schmid, C. (2012). Aggregating Local Image Descriptors into Compact Codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9).

Kendall, A. and Cipolla, R. (July 2017). Geometric loss functions for camera pose regression with deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584.

Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491.

Kendall, A., Grimes, M., and Cipolla, R. (2015). PoseNet: a convolutional network for real-time 6-DOF camera relocalization. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Kim, H., Dunn, E., and Frahm, J. (2017). Learned contextual feature reweighting for image geo-localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kim, H. J., Dunn, E., and Frahm, J.-M. (2015). Predicting Good Features for Image Geo-Localization Using Per-Bundle VLAD. In *Proc. ICCV*.

Kingma, D. P. and Ba, J. (2014a). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kingma, D. P. and Ba, J. (2014b). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kneip, L., Li, H., and Seo, Y. (September 2014). UPnP: An optimal O(n) solution to the absolute pose problem with universal applicability. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in neural information processing systems*, pages 109–117.

Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials (code). graphics.stanford.edu/projects/densecrf/densecrf.zip.

Krähenbühl, P. and Koltun, V. (2013). Parameter learning and convergent inference for dense random fields. In *International Conference on Machine Learning*, pages 513–521.

Krizhevsky, A., Nair, V., and Hinton, G. (2016). Cifar-10 (canadian institute for advanced research).

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Duerig, T., and Ferrari, V. (2018). The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv:1811.00982*.

Laddha, A., Kocamaz, M. K., Navarro-Serment, L. E., and Hebert, M. (2016). Map-supervised road detection. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 118–123.

Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81.

Li, H., Zhao, R., and Wang, X. (2014). Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification. *CoRR*, abs/1412.4526.

Li, Q., Arnab, A., and Torr, P. H. (2018a). Weakly-and semi-supervised panoptic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 102–118.

Li, X., Ylioinas, J., Verbeek, J., and Kannala, J. (September 2018b). Scene coordinate regression with angle-based reprojection loss for camera relocalization. In *Workshop track of Proceedings of European Conference in Computer Vision (ECCV)*.

Lian, X., Huang, Y., Li, Y., and Liu, J. (2015). Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2737–2745.

Lin, G., Shen, C., Van Den Hengel, A., and Reid, I. (2016). Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., and Reed, S. E. (2015a). SSD: single shot multibox detector. *CoRR*, abs/1512.02325.

Liu, W., Rabinovich, A., and Berg, A. C. (2015b). Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*.

Liu, X., Gao, J., He, X., Deng, L., Duh, K., and Wang, Y.-Y. (2015c). Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proc. NAACL*.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Long, M. and Wang, J. (2015). Learning multiple tasks with deep relationship networks. *CoRR*, abs/1506.02117.

Lowe, D. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2).

Ma, W.-C., Wang, S., Brubaker, M. A., Fidler, S., and Urtasun, R. (2016). Find your way by observing the sun and other semantic cues. *arXiv preprint arXiv:1606.07415*.

Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions. *Image and vision computing*.

Mishkin, D., Radenovic, F., and Matas, J. (2018). Repeatability Is Not Enough: Learning Affine Regions via Discriminability. In *Proc. ECCV*.

Mohan, R. (2014). Deep deconvolutional networks for scene parsing.

Muñoz-Bulnes, J., Fernandez, C., Parra, I., Fernández-Llorca, D., and Sotelo, M. A. (2017). Deep Fully Convolutional Networks with Random Data Augmentation for Enhanced Generalization in Road Detection. In *Submitted to the Workshop on Deep Learning for Autonomous Driving on IEEE 20th International Conference on Intelligent Transportation Systems*, Yokohama, Japan.

Naseer, T., Oliveira, G., Brox, T., and Burgard, W. (2017). Semantics-aware visual localization under challenging perceptual conditions. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Nickolls, J., Buck, I., Garland, M., and Skadron, K. (2008). Scalable parallel programming with cuda. In *ACM SIGGRAPH 2008 classes*, page 16. ACM.

Noh, H., Araujo, A., Sim, J., Weyand, T., and Han, B. (2017a). Large-Scale Image Retrieval with Attentive Deep Local Features. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:3476–3485.

Noh, H., Araujo, A., Sim, J., Weyand, T., and Han, B. (2017b). Large-scale image retrieval with attentive deep local features. In *IEEE International Conference on Computer Vision, (ICCV)*.

Noh, H., Araujo, A., Sim, J., Weyand, T., and Han, B. (2017c). Large-Scale Image Retrieval with Attentive Deep Local Features. In *Proc. ICCV*.

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528.

Odena, A., Dumoulin, V., and Olah, C. (2016). Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3.

Oliveira, G., Burgard, W., and Brox, T. (2016). Efficient deep methods for monocular road segmentation.

Paine, T., Jin, H., Yang, J., Lin, Z., and Huang, T. (2013). Gpu asynchronous stochastic gradient descent to speed up neural network training. *arXiv preprint arXiv:1312.6186*.

Papandreou, G., Chen, L., Murphy, K., and Yuille, A. L. (2015). Weakly- and semi-supervised learning of a DCNN for semantic image segmentation. *CoRR*, abs/1502.02734.

Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

Perd'och, M., Chum, O., and Matas, J. (2009). Efficient representation of local geometry for large scale object retrieval.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *Proc. CVPR*.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008a). Lost in quantization: Improving particular object retrieval in large scale image databases. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008b). Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases. In *Proc. CVPR*.

Pinheiro, P. O., Lin, T.-Y., Collobert, R., and Dollár, P. (2016). Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer.

Pylvänäinen, T., Roimela, K., Vedantham, R., Wang, R., and Grzeszczuk, R. (2010). Automatic alignment and multi-view segmentation of street view data using 3d shape priors. In *Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*.

Radenović, F., Iscen, A., Tolias, G., Avrithis, Y., and Chum, O. (2018). Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking. In *Proc. CVPR*.

Radenovic, F., Iscen, A., Tolias, G., Avrithis, Y., and Chum, O. (2018). Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5706–5715.

Radenović, F., Tolias, G., and Chum, O. (2016). CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples. In *Proc. ECCV*.

Radenović, F., Tolias, G., and Chum, O. (2018). Fine-tuning cnn image retrieval with no human annotation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Radwan, N., Valada, A., and Burgard, W. (2018). Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters*, 3(4):4407–4414.

Ranjan, R., Patel, V. M., and Chellappa, R. (2016). Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *CoRR*, abs/1603.01249.

Razavian, A. S., Sullivan, J., Carlsson, S., and Maki, A. (2016). Visual instance retrieval with deep convolutional networks. *ITE Transactions on Media Technology and Applications*.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

Ren, S., He, K., Girshick, R., and Sun, J. (2015a). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Proc. NIPS*.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015b). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Ronneberger, O., Fischer, P., and Brox, T. (2015a). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

Ronneberger, O., Fischer, P., and Brox, T. (2015b). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520.

Satkin, S., Lin, J., and Hebert, M. (2012). Data-driven scene understanding from 3D models. In *Proceedings of British Machine Vision Conference (BMVC)*.

Sattler, T., Zhou, Q., Pollefeys, M., and Leal-Taixé, L. (2019a). Understanding the limitations of cnn-based absolute camera pose regression. *CoRR*, abs/1903.07504.

Sattler, T., Zhou, Q., Pollefeys, M., and Leal-Taixe, L. (June 2019b). Understanding the limitations of cnn-based absolute camera pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Scharstein, D. and Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Schönberger, J. L., Pollefeys, M., Geiger, A., and Sattler, T. (2018). Semantic visual localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Schwing, A. G. and Urtasun, R. (2015). Fully connected deep structured networks. *CoRR*, abs/1503.02351.

Seeger, C., Müller, A., Schwarz, L., and Manz, M. (2016). Towards road type classification with occupancy grids. *IVS Workshop*.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229.

Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., and Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in RGB-D images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Siam, M., Mahgoub, H., Zahran, M., Yogamani, S., Jägersand, M., and Sallab, A. E. (2017). Modnet: Moving object detection network with motion and appearance for autonomous driving. *CoRR*, abs/1709.04821.

Siméoni, O., Avrithis, Y., and Chum, O. (2019). Local features and visual words emerge in activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11651–11660.

Simonyan, K. and Zisserman, A. (2014a). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proc. ICLR*.

Sivic, J. and Zisserman, A. (2003). Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proc. ICCV*.

Stewart, R., Andriluka, M., and Ng, A. Y. (2016). End-to-end people detection in crowded scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2325–2333.

Tang, M., Perazzi, F., Djelouah, A., Ayed, I. B., Schroers, C., and Boykov, Y. (2018). On regularized losses for weakly-supervised cnn segmentation. *arXiv preprint arXiv:1803.09569*.

Tao, R., Gavves, E., Snoek, C. G. M., and Smeulders, A. W. M. (2014). Locality in Generic Instance Search from One Example. In *Proc. CVPR*.

Teichmann, M., Araujo, A., Zhu, M., and Sim, J. (2019). Detect-to-retrieve: Efficient regional aggregation for image search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5109–5118.

Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., and Urtasun, R. (2018). Multinet: Real-time joint semantic reasoning for autonomous driving. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020. IEEE.

Teichmann, M. T. and Cipolla, R. (2019). Convolutional crfs for semantic segmentation. *British Machine Vision Conference (BMVC)*.

Tolias, G., Avrithis, Y., and Jegou, H. (2015a). Image search with selective match kernels: aggregation across single and multiple images. *International Journal of Computer Vision*, 116(3):247–261.

Tolias, G. and Jegou, H. (2014). Visual query expansion with or without geometry: refining local descriptors by feature aggregation. *Pattern Recognition*.

Tolias, G., Sicre, R., and Jégou, H. (2015b). Particular Object Retrieval with Integral Max-Pooling of CNN Activations. In *Proc. ICLR*.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656.

Torii, A., Arandjelović, R., Sivic, J., Okutomi, M., and Pajdla, T. (2015). 24/7 place recognition by view synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Torii, A., Havlena, M., and Pajdla, T. (2009). From google street view to 3d city models. In *Workshop in International Conference on Computer Vision (ICCV)*.

Triggs, B. and Verbeek, J. J. (2008). Scene segmentation with crfs learned from partially labeled images. In *Advances in neural information processing systems*, pages 1553–1560.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective Search for Object Recognition. *IJCV*, 104(2).

Vemulapalli, R., Tuzel, O., Liu, M.-Y., and Chellapa, R. (2016). Gaussian conditional random field network for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3224–3233.

Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y. (2014). Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393.

Weyand, T., Kostrikov, I., and Philbin, J. (2016). PlaNet - photo geolocation with convolutional neural networks. In *European Conference on Computer Vision (ECCV)*.

Wolcott, R. and Eustice, R. (2014). Visual localization within LIDAR maps for automated urban driving. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Wolpert, D. H., Macready, W. G., et al. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

Wu, Z., Shen, C., and van den Hengel, A. (2016). Wider or deeper: Revisiting the resnet model for visual recognition. *CoRR*, abs/1611.10080.

Wu, Z., Shen, C., and Van Den Hengel, A. (2019). Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133.

Xiao, T., Liu, Y., Zhou, B., Jiang, Y., and Sun, J. (2018). Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434.

Xu, D., Ouyang, W., Wang, X., and Sebe, N. (2018). Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 675–684.

Yi, K., Trulls, E., Lepetit, V., and Fua, P. (2016). LIFT: Learned Invariant Feature Transform. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Yim, J., Jung, H., Yoo, B., Choi, C., Park, D., and Kim, J. (2015). Rotating your face using multi-task deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 676–684.

Yu, F. and Koltun, V. (2015a). Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122.

Yu, F. and Koltun, V. (2015b). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer.

Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE.

Zhang, S., Zhang, C., You, Z., Zheng, R., and Xu, B. (2013). Asynchronous stochastic gradient descent for dnn training. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6660–6663. IEEE.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. (2014). Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer.

Zhang, Z., Rebecq, H., Forster, S., and Scaramuzza, D. (2016). Benefit of large field-of-view cameras for visual odometry. In *IEEE International Conference on Robotics and Automation, (ICRA)*.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015a). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. S. (2015b). Conditional random fields as recurrent neural networks. *CoRR*, abs/1502.03240.